Applying Dynamic Bayesian Networks in transliteration detection and generation

Peter Nabende
Promoter:  Prof. dr. ir. J. Nerbonne

Copromoter:  Dr. ir. E. T. K. Sang

Beoordelingscommissie:  Prof. dr. V. Baryamureeba
                        Prof. dr. M. Biehl
                        Prof. dr. K. U. Schulz
## Contents

**List of Figures** vii

**List of Tables** ix

**Abbreviations** xiii

1 Introduction 1
   1.1 Background ........................................ 1
   1.2 Research goal ..................................... 3
   1.3 Research approach ................................. 4
   1.4 Overview of the rest of the thesis ............... 5

2 A review on machine transliteration 9
   2.1 Introduction ........................................ 9
   2.2 Transliteration mining .............................. 10
      2.2.1 Data resources and transliteration mining methods ...... 11
      2.2.2 Discussion .................................... 18
   2.3 Transliteration generation ......................... 19
      2.3.1 Phonetic-based transliteration generation ............ 20
      2.3.2 Orthographic-based transliteration generation ........ 26
      2.3.3 Discussion .................................... 32
   2.4 Conclusion .......................................... 33
# Contents

## 3 Dynamic Bayesian Networks

3.1 Introduction ................................................. 35  
3.2 Bayesian networks ........................................ 36  
  3.2.1 Representation ....................................... 36  
  3.2.2 A transliteration example ............................ 38  
  3.2.3 Bayesian networks – inference ....................... 39  
3.3 Dynamic Bayesian Networks ............................... 40  
  3.3.1 DBNs – representation ............................... 40  
  3.3.2 DBNs – Inference .................................... 42  
  3.3.3 DBNs – Learning .................................... 42  
3.4 Conclusion ................................................. 45  

## 4 Pair HMMs for machine transliteration

4.1 Introduction ................................................. 47  
4.2 Hidden Markov Models ..................................... 48  
  4.2.1 A brief review on representation .................... 48  
  4.2.2 Recent use of HMMs in machine transliteration .... 49  
4.3 Pair Hidden Markov Models ............................... 50  
  4.3.1 Origins ............................................. 50  
  4.3.2 Pair HMMs for modeling word similarity ............ 52  
  4.3.3 Pair HMMs for modeling transliteration similarity 52  
  4.3.4 Pair HMMs – Inference .............................. 57  
  4.3.5 Pair HMMs – parameter estimation ................. 62  
4.4 Transliteration identification experiments using geographic names data .... 65  
  4.4.1 Data ................................................ 65  
  4.4.2 Evaluation setup and results ......................... 65  
4.5 Experiments using NEWS 2009 and 2010 shared task data .......... 75  
  4.5.1 Data ................................................ 75  
  4.5.2 Evaluation setup and results ......................... 77  
4.6 Conclusion ................................................. 84  

## 5 Transduction-based DBN models for machine transliteration

5.1 Introduction ................................................. 85  
5.2 Transduction-based DBN models ........................... 86  
  5.2.1 The memoryless stochastic transducer .............. 86  
  5.2.2 Representing the RY transducer as a DBN .......... 87  
  5.2.3 DBN templates for modeling transliteration similarity 90  
  5.2.4 Inference ......................................... 96  
5.3 Experiments (NEWS 2009 and 2010 shared task data) .......... 98
Contents

5.3.1 Data .................................................. 98
5.3.2 Evaluation setup and results ......................... 99
5.3.3 Error Analysis ........................................ 101
5.3.4 Computing transliteration similarity based on ensembles of DBN
models ....................................................... 101
5.4 Conclusion ............................................... 104

6 Applying DBN models in transliteration mining 105
6.1 Introduction ............................................... 105
6.2 Wikipedia – A source for transliteration mining .......... 106
   6.2.1 Transliteration mining using Wikipedia inter-language links .. 107
   6.2.2 Transliteration mining using comparable Wikipedia article text 109
6.3 DBN model selection for transliteration mining .................. 109
   6.3.1 Pair HMMs ........................................... 110
   6.3.2 Transduction-based context-dependent DBN models ........... 111
6.4 Experiments using NEWS 2010 shared task setup ............. 111
   6.4.1 Wikipedia inter-language link data ...................... 111
   6.4.2 Evaluation setup and results .......................... 113
6.5 Experiments using comparable Wikipedia article content ....... 121
   6.5.1 Extracting training data from Wikipedia ................. 121
   6.5.2 Comparable Wikipedia article content data .............. 122
   6.5.3 Evaluation setup and results .......................... 123
6.6 Conclusion ............................................... 125

7 Applying Pair HMMs in transliteration generation 127
7.1 Introduction ............................................... 127
7.2 Transliteration generation tasks .......................... 128
   7.2.1 Traditional machine transliteration task .................. 128
   7.2.2 Translating transliterations task ...................... 129
7.3 Using Pair HMMs in transliteration generation ............... 130
   7.3.1 Finite state automata ................................ 131
   7.3.2 Representing Pair HMMs as WFSTs ...................... 134
7.4 Experiments using NEWS 2009-2010 shared task data ........ 136
   7.4.1 Data ................................................. 136
   7.4.2 Transliteration models ................................ 137
   7.4.3 Evaluation metrics ................................... 140
   7.4.4 Results ............................................. 142
7.5 Experiments on translating transliterations ................... 145
   7.5.1 Data ................................................. 145
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Transliteration generation overview</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>A Bayesian network for relating different writing system characters</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>A 2-TBN and an unrolled network for a Hidden Markov model.</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Probabilistic version of pairwise alignment finite state automaton</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>Finite state representation of Pair HMM proposed by Mackay and Kondrak.</td>
<td>53</td>
</tr>
<tr>
<td>4.3</td>
<td>Finite state representation of a Pair HMM with 3 transition parameters.</td>
<td>55</td>
</tr>
<tr>
<td>4.4</td>
<td>Finite state representation of a Pair HMM with 9 transition parameters.</td>
<td>56</td>
</tr>
<tr>
<td>4.5</td>
<td>Finite state representation of the Random Pair HMM.</td>
<td>61</td>
</tr>
<tr>
<td>4.6</td>
<td>Variation of corpus cross entropy with test size for different Pair HMMs</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>on English-Russian Geonames corpus.</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Graphical representation for the MCI DBN template</td>
<td>89</td>
</tr>
<tr>
<td>5.2</td>
<td>A Pair HMM and a conceptual DBN representation for an alignment</td>
<td>91</td>
</tr>
<tr>
<td>5.3</td>
<td>A DBN representation of a Pair HMM with an illustrative assignment</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>of values to variables</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>Graphical representation for the memory DBN template</td>
<td>93</td>
</tr>
<tr>
<td>5.5</td>
<td>Graphical representation for the context-dependent DBN template.</td>
<td>94</td>
</tr>
<tr>
<td>5.6</td>
<td>Graphical representation for the context-dependent length DBN template.</td>
<td>95</td>
</tr>
<tr>
<td>6.1</td>
<td>Two Wikipedia articles on the same topic but in different languages</td>
<td>107</td>
</tr>
</tbody>
</table>
List of Figures

6.2 NEWS 2010 transliteration mining task overview. 108
6.3 Precision-Recall curves for Pair HMMs and a context-dependent DBN model after evaluation on mining English-Hindi transliteration pairs from Wikipedia. 118
6.4 Precision-Recall curves for Pair HMMs and a context-dependent DBN model after evaluation on mining English-Tamil transliteration pairs from Wikipedia. 120

7.1 An example of a finite state acceptor for accepting strings using a two symbol alphabet. 132
7.2 An FST for a standard Cyrillic romanization system. 133
7.3 A finite state transducer approximation of the Pair HMM with nine transition parameters. 135
7.4 An edit-distance based WFST. 138
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A Web-based MT system’s English to Chinese translation.</td>
<td>1</td>
</tr>
<tr>
<td>4.1</td>
<td>Transliteration data for four language pairs from the Geonames database.</td>
<td>65</td>
</tr>
<tr>
<td>4.2</td>
<td>Alphabet size per language in Geonames data.</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>Corpus cross entropy results for different Pair HMMs and algorithms on English-Russian Geonames corpus.</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>Number of iterations required to converge for different Pair HMM on Geonames data.</td>
<td>72</td>
</tr>
<tr>
<td>4.5</td>
<td>CVA and CVMRR transliteration identification results for two Pair HMMs and algorithms on English-French Geonames data.</td>
<td>73</td>
</tr>
<tr>
<td>4.6</td>
<td>CVA and CVMRR transliteration identification results for two Pair HMMs and different algorithms on English-Russian Geonames data.</td>
<td>73</td>
</tr>
<tr>
<td>4.7</td>
<td>CVA and CVMRR transliteration identification results for two Pair HMMs and different algorithms on English-Dutch Geonames data.</td>
<td>74</td>
</tr>
<tr>
<td>4.8</td>
<td>CVA and CVMRR transliteration identification results for two Pair HMMs and different scoring algorithms on English-German Geonames data.</td>
<td>74</td>
</tr>
<tr>
<td>4.9</td>
<td>Corpus size per language pair of NEWS 2010 shared task data.</td>
<td>76</td>
</tr>
<tr>
<td>4.10</td>
<td>Transliteration identification accuracy and MRR for baseline approach of pair n-gram models.</td>
<td>79</td>
</tr>
<tr>
<td>4.11</td>
<td>Transliteration identification accuracy for different Pair HMMs and scoring algorithms on English-Bangla NEWS 2010 shared task data.</td>
<td>80</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.12</td>
<td>Transliteration identification accuracy for different Pair HMMs and scoring algorithms on English-Chinese NEWS 2010 shared task data.</td>
<td>80</td>
</tr>
<tr>
<td>4.13</td>
<td>Transliteration identification accuracy for different Pair HMMs and scoring algorithms on English-Hindi NEWS 2010 shared task data.</td>
<td>81</td>
</tr>
<tr>
<td>4.14</td>
<td>Transliteration identification accuracy for different Pair HMMs and scoring algorithms on English-Kannada NEWS 2010 shared task data.</td>
<td>81</td>
</tr>
<tr>
<td>4.15</td>
<td>Transliteration identification accuracy for different Pair HMMs and scoring algorithms on English-Russian NEWS 2010 shared task data.</td>
<td>82</td>
</tr>
<tr>
<td>4.16</td>
<td>Transliteration identification accuracy for different Pair HMMs and scoring algorithms on English-Tamil NEWS 2010 shared task data.</td>
<td>82</td>
</tr>
<tr>
<td>4.17</td>
<td>Transliteration identification accuracy for different Pair HMMs and scoring algorithms on English-Thai NEWS 2010 shared task data.</td>
<td>83</td>
</tr>
<tr>
<td>5.1</td>
<td>Illustration of an alignment between a Russian word and an English word.</td>
<td>90</td>
</tr>
<tr>
<td>5.2</td>
<td>Transliteration identification accuracy and MRR for different DBN models.</td>
<td>100</td>
</tr>
<tr>
<td>5.3</td>
<td>Transliteration identification results examples for Error analysis.</td>
<td>102</td>
</tr>
<tr>
<td>5.4</td>
<td>Transliteration identification results for ensembles of DBN models.</td>
<td>103</td>
</tr>
<tr>
<td>6.1</td>
<td>Pair HMMs and transduction-based DBN models for mining transliterations from Wikipedia.</td>
<td>111</td>
</tr>
<tr>
<td>6.2</td>
<td>Number of Wikipedia topic pairs per language pair before and after data pre-processing.</td>
<td>113</td>
</tr>
<tr>
<td>6.3</td>
<td>A sample of English-Russian Wikipedia topic pairs.</td>
<td>114</td>
</tr>
<tr>
<td>6.4</td>
<td>A sample of expected transliteration mining results from Wikipedia topics.</td>
<td>114</td>
</tr>
<tr>
<td>6.5</td>
<td>NEWS 2010 shared task transliteration mining results on English-Russian data.</td>
<td>117</td>
</tr>
<tr>
<td>6.6</td>
<td>English-Hindi transliteration mining results for Pair HMMs and a context-dependent DBN model against NEWS 2010 shared task results.</td>
<td>119</td>
</tr>
<tr>
<td>6.7</td>
<td>English-Tamil transliteration mining results for Pair HMMs and a context-dependent DBN model against NEWS 2010 shared task results.</td>
<td>121</td>
</tr>
<tr>
<td>6.8</td>
<td>Wikipedia data for English and corresponding Russian articles.</td>
<td>124</td>
</tr>
<tr>
<td>6.9</td>
<td>Results for mining transliterations from English-Russian comparable articles.</td>
<td>125</td>
</tr>
<tr>
<td>7.1</td>
<td>Transliterated Russian names in four European languages.</td>
<td>129</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>7.2</td>
<td>Size of training, development, and testing datasets from NEWS 2009 and NEWS 2010 shared task per language pair.</td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>Transliteration generation results for three language pairs on NEWS 2010 shared task data.</td>
<td></td>
</tr>
<tr>
<td>7.4</td>
<td>Results for English→Russian transliteration generation on NEWS 2009 shared task data.</td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>LCSR and accuracy results for using WFSTs in translating translations.</td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>LCSR and accuracy results for using a PSMT system in translating transliterations.</td>
<td></td>
</tr>
<tr>
<td>7.7</td>
<td>A sample of results showing examples of some typical problems in translating transliterations with WFST and PSMT models.</td>
<td></td>
</tr>
</tbody>
</table>
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-TBN</td>
<td>Two-slice Temporal Bayes net</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>CCE</td>
<td>Corpus cross entropy</td>
</tr>
<tr>
<td>CLIR</td>
<td>Cross Language Information Retrieval</td>
</tr>
<tr>
<td>CON DBN</td>
<td>Context-dependent Dynamic Bayesian network</td>
</tr>
<tr>
<td>CPD</td>
<td>Conditional probability distribution</td>
</tr>
<tr>
<td>CPT</td>
<td>Conditional probability table</td>
</tr>
<tr>
<td>CRF</td>
<td>Conditional Random Field</td>
</tr>
<tr>
<td>CVA</td>
<td>Cross validation accuracy</td>
</tr>
<tr>
<td>CVMRR</td>
<td>Cross validation mean reciprocal rank</td>
</tr>
<tr>
<td>DBN</td>
<td>Dynamic Bayesian network</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation maximization</td>
</tr>
<tr>
<td>FSA</td>
<td>Finite state acceptor</td>
</tr>
<tr>
<td>FST</td>
<td>Finite state transducer</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>GEM</td>
<td>Generalized expectation maximization</td>
</tr>
<tr>
<td>GMTK</td>
<td>Graphical modeling toolkit</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov model</td>
</tr>
<tr>
<td>LCSR</td>
<td>Longest common subsequence ratio</td>
</tr>
<tr>
<td>MAP</td>
<td>Mean average precision</td>
</tr>
<tr>
<td>MCI DBN</td>
<td>Memoryless and context-independent Dynamic Bayesian network</td>
</tr>
<tr>
<td>MEM DBN</td>
<td>Memory-dependent Dynamic Bayesian network</td>
</tr>
<tr>
<td>MRR</td>
<td>Mean reciprocal rank</td>
</tr>
<tr>
<td>MT</td>
<td>Machine Translation</td>
</tr>
<tr>
<td>NE</td>
<td>Named entity</td>
</tr>
<tr>
<td>NEWS</td>
<td>Named entities workshop</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>OCR</td>
<td>Optical character recognition</td>
</tr>
<tr>
<td>OOV</td>
<td>Out of vocabulary</td>
</tr>
<tr>
<td>Pair HMM</td>
<td>Pair Hidden Markov model</td>
</tr>
<tr>
<td>PGM</td>
<td>Probabilistic graphical model</td>
</tr>
<tr>
<td>PSMT</td>
<td>Phrase-based Statistical Machine Translation</td>
</tr>
<tr>
<td>SMT</td>
<td>Statistical Machine Translation</td>
</tr>
<tr>
<td>TI</td>
<td>Transliteration identification</td>
</tr>
<tr>
<td>TM</td>
<td>Transliteration mining</td>
</tr>
<tr>
<td>WFSA</td>
<td>Weighted finite state acceptor</td>
</tr>
<tr>
<td>WFST</td>
<td>Weighted finite state transducer</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

With the advent of the Web, Natural Language Processing (NLP) systems such as Machine Translation (MT) are increasingly being accessed and used for cross language information processing. NLP systems are useful as they help to overcome various limitations that are initially associated with manual information processing. Currently, a major benefit of using NLP systems is the instant generation of output given input data, and hence the possibility of processing and handling large amounts of data even with low cost computational resources. However, gains in processing capability of NLP systems are offset by poor output quality. There are various factors that can affect system output quality. In this thesis, the concern is with the case where NLP systems encounter words or phrases in data that are unknown to them.

Most NLP systems loose output quality when encountered with ‘new’ words. In NLP applications such as MT and Cross Language Information Retrieval (CLIR), the unknown words are commonly referred to as Out of vocabulary (OOV) words. In cross-language applications, the problem caused by the unknown words is further complicated by a lack of a corresponding representation in some target language. Table 1.1 illustrates this problem where a state-of-the-art Web-based machine translation system (Google Translate\(^1\)) fails to get corresponding representations in the

<table>
<thead>
<tr>
<th>English input</th>
<th>Chinese translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>While the battle against malaria is gradually being won according to Dr. Falade, the use of Fansidar as a combination drug is highly discouraged.</td>
<td>儘管防治瘧疾的戰鬥正在逐漸獲得勝利，根據博士Falade，使用的Fansidar作為一個組合藥物是非常氣餒。</td>
</tr>
</tbody>
</table>

Table 1.1: A Web-based MT system’s English-to-Chinese translation

---

\(^1\)http://translate.google.com
Chinese language for the two underlined words (*Falade* and *Fansidar*) written using the Latin alphabet and in an English sentence. As Table 1.1 shows, the strategy used by the MT system and which is commonly used by similar systems in dealing with unknown words is to simply copy them to the resulting output. Table 1.1 also shows that all the words that the system failed to represent in the Chinese translation are names (*Falade* is a person name while *Fansidar* is a drug name). This strategy of simply copying unknown words may be useful in cases where the source and target language use the same writing system since there is a likelihood to retain spellings for named entities across the languages. When the source and target language use different writing systems, the strategy of copying unknown words is not useful.

Transliteration, a process used to convert new words from a source language to a phonetically equivalent, understandable, and representable form using the writing system of the target language is currently the most natural approach to dealing with unknown words for the case where different writing systems are used. For example, a suggestion of a phonetically equivalent representation for the word *Fansidar* in Table 1.1 could be like this in Chinese: 反思大 /fan-si-da/. In this case, the main NLP system would require an additional transliteration sub-system that helps generate hypothetical target language representations for any identified unknown word. A different approach to employing transliteration in a cross-language processing system is to complement the system’s bilingual lexicon with a separately acquired transliteration pair lexicon. Either way, transliteration is currently important both as a topic and as a sub-task in NLP since system effectiveness is expected to increase when it is used.

In both of the transliteration-based approaches in the last paragraph, various methods have been proposed and used to help improve the quality of the system generated transliterations or transliteration-pair lexicons. However, recent work, for example the shared tasks on transliteration mining (Kumaran et al. 2010b) and transliteration generation (Li et al. 2010), shows that there is need to identify more approaches that can help improve system performance in the two tasks. Research on using a given method in each of the two transliteration-related tasks is usually based on an interest of attaining improvements in system performance.

In this thesis, the interest is to see whether models derived from two Dynamic Bayesian Network (DBN)-related approaches can lead to improvements in transliteration mining and generation. The two DBN-related approaches are: Pair Hidden Markov Models (Pair HMMs) and a transduction-based DBN framework. The Pair HMM approach originates from work in biological sequence analysis (Durbin et al. 1998) and was later adapted to compute word similarity and successfully applied in cognate identification (Mackay and Kondrak 2005, Kondrak and Sherif 2006) and in dialect comparison (Wieling et al. 2007). The transduction-based DBN framework
1.2 Research goal

The overall research goal in this thesis is to apply Pair Hidden Markov models (Pair HMMs) and transduction-based Dynamic Bayesian Network models in transliteration mining and generation while aiming for improvements over existing techniques. Based on this goal, the thesis aims to address the following questions:

1. Does the current state of research necessitate an investigation into the use of new methods (such as the Dynamic Bayesian Network (DBN) models proposed in this thesis) for transliteration mining and generation?
   For this question, we would like to know whether existing methods for transliteration mining and generation suffice.
2. Can DBN models that have been used in tasks (such as cognate identification and pronunciation classification) with requirements similar to transliteration mining be valuable when used in the context of computing transliteration similarity? Related to that, can modifications to these DBN models that meet the requirements for computing transliteration similarity be valuable in the identification of transliteration pairs?

Here, we would like to know whether the assumptions associated with the successful application of DBN models in previous tasks could also lead to successful application of the DBN models in transliteration detection and generation.

3. What features are critical to the use of DBNs for modeling transliteration similarity?

In the thesis, we investigate various model generalizations with respect to structure and parameter settings. We would like to know which types of Pair HMMs and transduction-based DBN models adequately address factors that are important in modeling transliteration similarity.

4. Can the application of DBN models improve transliteration mining and generation quality as compared to current state-of-the-art methods?

Results obtained from representative experimental setups may not portray the true effect of applying DBN models on real-world data. Here, we are interested in knowing whether there can be any benefits of using DBN models when evaluated on real-world data for transliteration detection and generation. This, as the question has been put, requires an evaluation against state-of-the-art methods that would be applied in a similar manner.

1.3 Research approach

The research approach used in this thesis is mostly focused on addressing the questions in the previous section. To address the first and second questions, I undertook an exhaustive literature review on approaches that have been proposed and used in previous work for transliteration mining and generation. In order to avoid repetitions, I also considered recent comprehensive literature reviews on machine transliteration, for example in Karimi et al (2011). The literature review is mostly exploratory, but a critical analysis is given where suitable.

To address the remaining questions, we follow an empirical approach in which experiments are conducted to evaluate the performance of several DBN models in transliteration detection and generation. For the third and fourth question, we define a transliteration identification (TI) experimental setup to evaluate the use of
proposed DBN models for computing transliteration similarity. In the TI experimental setup, we first experiment with our own prepared transliteration data from a Web-based geographic names database (Geonames) and later we experiment with standard transliteration corpora from the 2009 and 2010 shared tasks on transliteration generation (Li et al. 2009, Li et al. 2010). Each dataset has been manually verified and we assume that each source language word has exactly one target language word match. We use the TI experimental setup at this stage to identify DBN models that could be useful for transliteration generation and in detecting transliteration pairs from real-world ‘noisy’ data.

For the fourth research question, our participation in both the 2009 and 2010 NEWS shared tasks on transliteration generation and transliteration mining respectively ensured an evaluation of the application of the proposed DBN models against state-of-the-art methods that were also applied on the same standard transliteration corpora. In the thesis, I have followed the same evaluation setups as specified for the NEWS 2009 shared task on transliteration generation (Li et al. 2009) and for the NEWS 2010 shared task on transliteration mining (Kumaran et al. 2010b) for evaluating the DBN models. For the transliteration generation task, participating teams were supplied with training and development data for a dozen language pairs to be used for training and tuning the participating systems. After training and development, each participating team was required to submit ten system generated candidate transliterations per source language word in the test set for a given language pair. For the transliteration mining task, each participating team was availed with a seed set of matching name pairs for five language pairs to be used as initial training data. In these experiments we use models that performed well in the experimental transliteration identification setup mentioned in the previous paragraph.

1.4 Overview of the rest of the thesis

In Chapter 2, we present a literature review on transliteration mining and generation aimed at determining the current state of research on the two tasks and the need for new solutions to some identified gaps. We give a general view for each of the two tasks and a description of the main phases. We then review several modeling approaches that have been used in each task ranging from the earliest to the current state-of-the-art. This also involves an analysis of performances achieved by the approaches in the two tasks.

In Chapter 3, we introduce the main concepts underlying the framework of Dynamic Bayesian Networks with respect to three aspects: their representation, DBN inference methods, and DBN learning methods. For DBN learning in particular, we review a theoretical explanation of the expectation maximization (EM) algorithm.
The EM algorithm and its generalized form are applied in different ways to train all DBN models that we have proposed for transliteration mining and generation. At the end of the chapter, we specify the general framework for applying the DBN models in the two transliteration mining and transliteration generation tasks.

In Chapter 4, we introduce the Pair HMM approach as the first of the DBN methods proposed for use in transliteration mining and generation. First, we provide some background regarding the origins of the Pair HMM method from its inception in the field of biological sequence analysis (Durbin et al. 1998) to its adaptation for estimating word similarity (Mackay and Kondrak 2005). A discussion then follows of the requirements that need to be met in order to adapt the Pair HMM approach for estimating transliteration similarity. Different plausible Pair HMM parameterization settings are proposed and evaluated in an experimental transliteration identification (TI) setup. We first investigate two settings for the Pair HMMs in the experimental TI task: in the first setting, we assume that only one vocabulary is used to generate the source and target words, and in the second setting, we assume that the Pair HMMs use separate vocabularies corresponding to the source and target language writing systems. It is quite obvious that the second setting relates more with transliteration, and the experiments are aimed at determining the necessity to capture the differences in the source and target language vocabularies in Pair HMM emission parameters for transliteration similarity estimation. We consider the use of the TI setup for two cases: one case where the source and target language use the same writing system (for example between English and Dutch), and one case where the source and target language use different writing systems (for example between English and Russian). The datasets used at this stage are obtained from the Geonames database. We also investigate the use of different Pair HMM string similarity scoring algorithms and the the use of different definitions for Pair HMM transition parameters. Here, we use standard corpora from the NEWS 2009 and NEWS 2010 shared tasks on transliteration generation. The transliteration identification performance of the Pair HMM approach is evaluated against that of a standard baseline approach of using ‘pair n-gram’ models.

In Chapter 5, we introduce the transduction-based Dynamic Bayesian Network approach as the second DBN-related approach we have proposed to apply for computing transliteration similarity. First, we review the approach as initially proposed by Filali and Bilmes (2005) in their pronunciation classification task. A discussion then follows regarding the adaptation of the transduction-based DBN modeling approach in the context of transliteration similarity estimation. In proposing different transduction-based DBN model generalizations, we start with a presentation of an approximate transduction-based DBN model representation for the Pair HMMs. We have successfully adapted three DBN model generalizations associated with the
transduction-based DBN modeling approach for computing transliteration similarity. These are described in addition to the baseline DBN model template from which they are derived. Each of the DBN model generalizations is used to account for specific types of temporal dependencies, including: dependencies that capture memory from previous edit states of a DBN model; contextual dependencies of edit states of a DBN model on either source or target string elements or on elements from both source and target strings; and dependencies that account for the lengths of edit steps needed for string similarity estimation. We then investigate the use of several models from the DBN model generalizations in the experimental transliteration identification task introduced in Chapter 4 using standard transliteration corpora from the shared tasks as mentioned in Chapter 4. The performance from the use of the transduction-based DBN models is evaluated against that of the baseline pair n-gram approach and the best performing Pair HMMs in Chapter 4. Our analysis of the results leads us to further propose and test several ensembles of DBN models for computing transliteration similarity.

In Chapter 6, we present an evaluation of the use of the DBN models in mining transliterations from real-word data (specifically, from the Web-based Wikipedia encyclopedic resource). Two transliteration mining sub-tasks for evaluating the DBN models are first introduced. In the first sub-task, we use the same evaluation set-up as that used in the NEWS 2010 shared task on transliteration mining where participating systems are evaluated on mining transliterations from paired cross-language Wikipedia topics. For this task, we evaluate the Pair HMM and transduction-based DBN methods against state-of-the-art methods that were used by the other participating teams in the shared task. In the second sub-task we propose and evaluate the application of the DBN models on paired cross-language Wikipedia article content in addition to using the respective paired Wikipedia topics.

In Chapter 7, we present an evaluation of the use of Pair HMMs in transliteration generation. Although Pair HMMs were initially proposed just for the purpose of computing string similarity – where they have been successfully used – this chapter is aimed at determining whether the Pair HMMs could as well be valuable in transliteration generation. Two transliteration generation sub-tasks are first introduced. In the first sub-task, we use the same evaluation setup as that used in the NEWS 2009 (Li et al. 2009) and NEWS 2010 (Li et al. 2010) shared tasks on transliteration generation. In the second sub-task we propose using the transliteration generation framework for translating named entities between languages that use the same writing system. We then describe a scheme for representing Pair HMM parameters as parameters in weighted finite state automata to allow for their use in transliteration generation. We also describe various types of other weighted finite state automata for evaluation in addition to the Pair HMM-based models. For the first task, we report
on results associated with the use of weighted finite state automata including the Pair HMM-based models and compare them to results associated with the use of a state-of-the-art phrase-based statistical machine translation approach. For the second task, we evaluate the weighted finite state automata models and the phrase-based statistical machine translation approach against the standard baseline of copying unknown words.

Chapter 8 concludes the thesis with a discussion of results on the application of the two proposed DBN-related approaches in the two tasks of transliteration mining and generation. We also point out the contributions of the thesis. Finally, we present our suggestions for future work including work that we have not managed to cover in the thesis.
Chapter 2

A review on machine transliteration

2.1 Introduction

The origins of the use of transliteration as a process for dealing with unknown words in a foreign language or in a dialect of a particular language seem to be non-existent in transliteration literature. However, the systematic attempts to create systems for representing characters in a writing system of origin (for example in Japanese or Chinese) to characters in a different language using its writing system (for example to English using the Latin alphabet) are well documented and these systems are commonly referred to as ‘transliteration systems’. The term Romanization is often associated with transliteration systems where the Roman alphabet is used to represent characters from a different writing system. For example, the American Library Association-Library of Congress (ALA-LC) romanization tables\(^1\) constitute one of the largest collection of romanization systems for representing text in more than 150 languages written in various non-Roman scripts using the Latin alphabet. With regard to words, contemporary literature suggests that transliteration is likely to have started as a process for converting a given word in a language of origin to a phonetically equivalent, understandable, and orthographically representable form in some target language (Li et al. 2009). More strictly, if the conversion is from a language of origin, the process is called forward transliteration. Backward transliteration or reverse transliteration is defined as the reverse process, where the aim is to find the original word representation in a language of origin given an existing word in a foreign language (for example finding the original Russian name (дмитрий) given an English name variant ‘Dmitriy’).

The use of automated Natural Language Processing (NLP) applications such as Machine Translation (MT) and the non-diminishing importance of out of vocabulary (OOV) words that these applications encounter necessitated the use of automated

\(^1\)http://www.loc.gov/catdir/cps0/roman.html
methods as well to help deal with them. The most popular reference to one of the earliest attempts at fully automating the process of transliteration in the sense of processing named entities dates back to almost two decades ago (Arbabi et al. 1994) where a combination of a rule-based expert system and an artificial neural network were used for including vowels in Arabic names. Also, around just the same time, studies on the automated search for named entities within a language and across languages had already started. Currently the general term used for cross-language named entity search for the case where the languages use different writing systems is transliteration mining. From the later half of the 1990s on, various approaches have been proposed to handle named entities (NEs) in cross language applications with regard to both the converting of NEs from one writing system to another, and to the search of corresponding NEs in different writing systems. In this chapter, we introduce the current view of the two tasks: transliteration generation, and transliteration mining, and with respect to each task, we review some of the major approaches that have been used from the earliest to current state-of-the-art. The organization of our review will follow the same order of presentation for the two tasks throughout the thesis.

**Notation**

To simplify our review of the various transliteration mining and generation approaches, we establish some notation that is common to most of the methods. The transliteration process in both mining and generation involves an analysis of the source and target language words which we denote here as: $S$ for a source language word and $T$ for a target language word. However, we shall extend the notation whenever there is need to reflect the point of discussion. For example, it may be necessary to specify a source word constituting $m$ characters as $S^m_1 = s_1 s_2 ... s_m$ and a target word with $n$ characters as $T^n_1 = t_1 t_2 ... t_n$. When discussing a phonetic-based approach we use $SP$ to denote the phonetic representation of a source word, while $TP$ is used to denote the phonetic representation of the target word. For the constituent phonetic units, we use $SP^i_1 = sp_1 sp_2 ... sp_i$ for the source word and $TP^k_1 = tp_1 tp_2 ... tp_k$ for the target word. For other specific representations, additional notation will be defined as per the need.

### 2.2 Transliteration mining

The process of mining transliterations generally involves the search for corresponding NEs from a collection of candidate NEs between two or more languages in different writing systems. The main differences in the transliteration mining approaches are
associated with: the data source for obtaining candidate NEs in each of the languages; the methods that are used to identify candidate NEs; and the methods that are used to relate and extract transliteration pairs. We shall review transliteration mining approaches based on the type of data resource used. For each type of data resource, we present some examples from transliteration literature and the subsequent methods that are used for candidate NE identification, transliteration similarity estimation, and transliteration pair extraction.

2.2.1 Data resources and transliteration mining methods

Transliteration mining necessitates the use of a bi(multi)-lingual\(^2\) corpus from which we expect to match bi-lingual NEs. That is, the resource when considered as a whole, should have a reasonable amount of text in at least two or more languages to enable the identification and extraction of similar words across different languages. The most common types of bi-lingual data resources for transliteration mining can be classified into: bi-lingual single document texts, which consist of texts in two or more languages in the same document; parallel corpora, which consists of texts in two or more languages and where the texts are translations of one another (Karimi et al. 2011) and in different documents where corresponding sentences that are related through a given identifier are exact translations of each other; and comparable corpora, which is text in two or more languages and in different documents where the corresponding text are not exact translations of each other (as is the case in parallel corpora). The main differences in the transliteration mining process for the different approaches are associated with the kind of data resource that is used and the identification of candidate NEs from a given data resource. After the identification of candidate NEs, the setup for comparing and extracting NEs across different languages is often similar. In the following we review a selection of some examples and transliteration mining methods associated with each of the different types of bi-lingual data resources.

a) Bi/Multi-lingual single document text

The use of single document text for mining transliteration pairs usually requires the application of prior knowledge about the presentation of different entities in a given document. Based on how bi-lingual text is represented, the transliteration mining process may be simplified or may require some additional pre-processing steps before

\(^2\)Although the term multi-lingual is a generalization of bi-lingual, most of the approaches utilize bilingual resources. We therefore prefer to use the term bi-lingual in a general discussion to represent both cases. However, when describing a given approach that uses multi-lingual resources (that is, in more than two languages), we will correctly specify it as a multi-lingual resource.
applying a transliteration similarity estimation method. In the examples below, we see two different representations. In one example, source words are hypothesized to exist in parentheses next to target words in a sentence (Lin et al. 2004). In the other example (Kuo et al. 2007), source words are hypothesized to collocate with target words in a sentence but there is rarely an existence of delimiters that enclose source words.

Kuo et al. (2007) use predominantly Chinese Web pages, where transliterated words are collocated closely with their original source (English) words and the source words are often appositives of neighboring target (Chinese) words in a close context. They assume that the scope of a close context is within a sentence boundary which is delimited by punctuation such as full stops, commas, question and exclamation marks; and that it is a range of proximity where a source word and its target transliteration collocate. Kuo et al. also suggest that in cases where there are different types of words in a close context, we need to consider only word pairs that are most likely to be associated with phonetic transliteration. To describe how candidate named entities are identified, we use one of their examples, which illustrates collocations of source and target language words:

"...經營Kuro 庫洛P2P 音樂交換軟體的飛行網, 3日發表P2P與版權爭議的解決方案C2C (Content to Community)...

In the example above, “庫洛 /KU-LUO/” is a transliteration for “Kuro”, the two can also be qualified as a candidate transliteration pair. However, although C2C is collocated with “Content to Community”, the latter is just an acronym expansion and not a transliteration; such a pair cannot be used as a candidate transliteration pair. Based on this observation, Kuo et al. propose a procedure for identifying candidate pairs which is as follows: 1) the predominantly Chinese Web page is segmented into sentences using punctuation marks as delimiters; 2) a search is made for any source language word $S$ in each sentence; 3) if an English word $S$ is recognized, then a $k$-neighborhood is defined to serve as the close context of the recognized English word; 4) $T \in \Omega$ is defined as a target transliteration candidate in the $k$-neighborhood, where $\Omega$ is the set of all transliteration candidates in the $k$-neighborhood. In the example above, “Kuro” is recognized as an English word, “經營 /JIN-YIN/” and “庫洛 /KU-LUO/” are suggested in a close context, the left and right $k$-neighborhoods. Two candidate pairs, “Kuro-經營” and “Kuro-庫洛” are then selected for further examination. For transliteration similarity estimation, Kuo et al. (2007) use a phonetic similarity (PS) modeling approach. The candidate words are first transformed into syllabic sequences $S_{sy}$ for the English word and $T_{sy}$ for the Chinese word. The PS model is then used to identify the most probable $T_{sy}'$ that matches $S$. In the PS model, they formulate their transliteration process using the noisy channel model (Brown et al. 1993) and
2.2 Transliteration mining

by applying the Bayesian rule, \( P(T|S) \) is expressed as:

\[
P(T|S) = \frac{P(S|T)P(T)}{P(S)} \tag{2.1}
\]

where \( P(S|T) \) represents the noisy channel probability (also called transliteration probability) and \( P(T) \) is the language model probability. \( P(S|T) \) is approximated using a phonetic confusion probability \( P(S_{sy}|T_{sy}) \) which is obtained from a phonetic confusion probability matrix. They propose three ways of estimating the syllable-based confusion matrix: 1) \( P_{ASR}(S_{sy|m}|T_{sy|n}) \) for which an automatic speech recognition (ASR) system is used and where a labeled English speech database is run through a Chinese ASR system; 2) \( P_{SYL}(S_{sy|m}|T_{sy|n}) \) for which a Syllable PSM is used and where the syllable confusion probability is estimated by extracting transliteration pairs which are converted to syllables and to phonemes; 3) \( P_{SS}(S_{sy|m}|T_{sy|n}) \) for which a Sub-syllable PSM is used and where the syllable confusion matrix is estimated using sub-syllable confusion probability. Kuo et al. exploit the three confusion matrices in different stages for transliteration similarity estimation. After obtaining a similarity score and ranking the candidate list of \( T \), they identify the most probable \( T' \) by using a hypothesis test to decide whether \( T' \) is a transliteration of \( S \).

Lin et al. (2004) use both single document bi-lingual text and parallel text for extracting English Chinese transliterations. We present their approach for the single document text here and that for the parallel text in the next subsection on Parallel corpora. For the single document bi-lingual text, they exploit the fact that some data resources print source language terms in parentheses following their transliterations as shown in their example below:

國西部城市(1995年人口約247,000),位於科隆(Cologne)西北方...

In the example above 科隆 is a transliteration of Cologne. During transliteration similarity estimation, Lin et al. use a statistical transliteration model. The source language word \( (S) \) is first split into \( k \) transliteration units (TU:s) \( S = su_1, su_2, ..., su_k \) which are then converted independently into \( k \) target characters \( tc_1, tc_2, ..., tc_k \) using the statistical transliteration model. The \( tc_j \)'s are subsequently combined to produce a target word \( T \). Their transliteration model for \( P(tc_j|su_i) \) is estimated using an Expectation Maximization (EM) algorithm with Viterbi decoding.

\(^3\)Kuo et al. (2007) provide a much more detailed description of their approach with many mathematical formulations, but because of space constraints, we have decided to omit most of the details.
b) Parallel corpora

The use of parallel corpora necessitates the alignment of sentences between two languages. As will be seen below, some approaches apply a sentence alignment procedure as part of the transliteration mining process while other approaches use data resources where the sentences are already aligned. Each aligned sentence pair is then hypothesized to contain similar NEs across the different languages. Most of the approaches go a step further in filtering out unnecessary entities in at least one of the languages before applying a transliteration similarity estimation method.

Sherif and Kondrak (2007a) use two sets of sentence aligned bi-text from an Arabic tree bank part 1-10k word English translation corpus, and an Arabic English parallel news Part 1 corpus. They report that the two corpora contain Arabic news articles and their English translations aligned at the sentence level. They use the tree bank data as development data to optimize the acceptance threshold used by one of their methods for transliteration similarity estimation and extraction. They use the following pre-processing procedure to identify candidate NEs. First, they tokenize the English corpus using a tokenization tool (Word splitter). After tokenization, they remove all uncapitalized words; stop words are also removed from both sides of the bi-text. Lastly, English words of length less than 4 and Arabic words of length less than 3 are removed. Sherif and Kondrak (2007a) then apply a number of models for word similarity estimation including the bootstrapped stochastic transducer which is their main proposed method in the paper. Below, we summarize the similarity estimation methods that Sherif and Kondrak use:

1) Levenshtein edit distance (LED). Is used as the baseline method. To enable the computation of LED, a common representation between the source and target languages is needed. Specifically, Arabic candidate NEs are romanized to get to the common representation.

2) ALINE, is a phonetic-based word similarity estimation algorithm where individual phonemes input to the algorithm are decomposed into a dozen phonetic features, such as Place, Manner, and Voice. Then, a substitution score between a pair of phonemes is based on the similarity as assessed by a comparison of the individual features. After an optimal alignment of the two words is computed with a dynamic programming algorithm, the overall similarity score is set to the sum of the scores of all links in the alignment normalized by the length of the longer of the two words. The source and target words are first converted into phonetic transcriptions using a deterministic rule-based transformation.

3) Fuzzy string-matching algorithm. This was initially proposed by Freeman et al. (2006). The Fuzzy matching algorithm is based on the Levenshtein Edit-Distance but encodes more knowledge about the relationships between the source and target
2.2 Transliteration mining

In this case, the LED is computed using letter equivalences as matches instead of identities. The source and target language letters within a class are treated as identities. The resulting Levenshtein distance is then normalized by the sum of the lengths of both words.

4) The main model proposed in Sherif and Kondrak (2007a) is a stochastic transducer from Ristad and Yianilos (1997) which is trained iteratively, and then applied to score a pair of candidate NEs.

In a rather different NE identification approach, Lee and Chang (2003) first apply a sentence alignment procedure to align parallel texts at the sentence level. An NE tagger is used to identify proper nouns in the source text (English) which serve as candidate source NEs for identifying transliterations in the target language (Chinese). Lee and Chang (2003) also formulate the transliteration problem as a noisy channel model while exploiting phonetic similarities between source words (S) and target words (T). The computation for \( P(T|S) \) is first formulated as a marginalization over an alignment sequence (\( \delta \)):

\[
P(T|S) = \sum_{\delta} P(T, \delta|S) = \sum_{\delta} P(T|\delta, S)P(\delta|S).
\]  

(2.2)

where \( \delta \) represents an alignment candidate with \( \delta = \delta_1, \delta_2, ..., \delta_N \) match types. To reduce computational complexity, the summation criterion in Equation 2.2 is changed into a maximization and \( P(T|S) \) is approximated as:

\[
P(T|S) \approx \max_{\delta} P(T|\delta, S)P(\delta|S) \approx \max_{\delta} P(T|\delta, S)P(\delta) \quad (2.3)
\]

Using transliteration units \( su_1^N \) for the source word and \( tu_1^N \) for the target word, Lee and Chang re-approximate \( P(T|\delta, S)P(\delta) \) in Equation 2.3 as follows:

\[
P(T|\delta, S)P(\delta) = P(tu_1^N|su_1^N)P(\delta_1, \delta_2, ..., \delta_N) \approx \prod_{i=1}^{N} P(tu_i|su_i)P(\delta_i). \quad (2.4)
\]

Finally \( \log P(T|S) \) is computed as:

\[
\log P(T|S) \approx \max_{\delta} \sum_{i=1}^{N} (\log P(tu_i|su_i) + \log P(\delta_i))
\]

The maximum accumulated log probability among all possible alignment paths is computed using a dynamic programming procedure. Lee and Chang (2003) estimate the model probabilities using an Expectation Maximization (EM) procedure. They also incorporate some linguistic processing in their method, first to accelerate the convergence of EM training and then during transliteration similarity estimation to improve transliteration identification quality.
Lin et al. (2004) use a named entity identification approach similar to Lee and Chang’s (2003) approach above. We already saw in the previous section (on Bi/Multi-lingual single document text) that Lin et al. (2004) use both the single document bilingual text and parallel text to mine transliteration pairs. For parallel text, Lin et al. first identify proper names in the source (English) sentence and then subsequently identify transliterations for each proper name. They suggest the use of a part of speech tagger and named entity recognizer for identifying English proper nouns. All words in the target language (Chinese) sentence are considered as transliteration candidates. They then use Viterbi decoding to identify the transliterations in the target language sentence using the same procedure as described at the end of the previous subsection.

c) Comparable corpora

Comparable corpora is mainly obtained from Web-based cross-language text, usually with the aid of cross-language links. Common sources include: time-aligned news articles over a given period of time, and encyclopedic resources such as Wikipedia. Most of the recent work in transliteration mining use comparable corpora as they are easy to acquire, and they can be used without the need of applying a sentence alignment procedure as is the case for parallel corpora. In this case, we simply use similar text with the assumption that they are likely to contain a reasonable number of corresponding NEs between the source and target languages. In the following, we review some of the recent approaches that use comparable corpora.

Udupa et al. (2009) use comparable news corpora for mining NE transliteration equivalents. They investigate the effectiveness of using different sets of paired documents for mining transliteration equivalents. In one stage, they investigate the use of documents comprising the comparable corpora \((C_s, C_t)\), and in another stage, they investigate the use of paired documents \((A_s, A_t)\) obtained from \((C_s, C_t)\) as highly similar documents. In the latter stage, they use a cross-language document similarity model (KL-divergence) to estimate content similarity between documents \((D_s, D_t)\). Using \(V_s\) denotes the source language vocabulary and \(V_t\) the target language vocabulary, Udupa et al. compute KL-divergence as follows:

\[
-KL(D_s\|D_t) = \sum_{T \in V_t} P(T|D_s) \log \frac{P(T|D_t)}{P(T|D_s)}
\]

and since the interest is in target documents which are similar to a given source document, the numerator in Equation 2.5 (which is independent of the target document) can be ignored. By expanding \(P(T|D_s)\), the following Equation is used for computing cross-language similarity:
2.2 Transliteration mining

\[
\sum_{T \in V_t} \sum_{S \in V_s} P(S|D_s)P(T|S) \log P(T|D_t).
\]

Udupa et al. then use two transliteration similarity models on each document pair \((D_s, D_t)\) in \(A_{s,t}\) to produce a set \(\text{Pairs}_{s,t}\) of NE transliteration equivalents. The first model, which they call the discriminative transliteration model uses a logistic function to compute transliteration similarity between every candidate pair of words \((S, T)\) in each \((D_s, D_t)\):

\[
\text{Transliteration similarity } (S, T, \theta) = \frac{1}{1 + e^{-w \cdot \phi(S, T)}}
\]  

(2.6)

where \(\phi(S, T)\) is the feature vector for the pair of words \(\phi(S, T)\) and \(w\) is the weights vector which is learnt discriminatively using a bi-lingual list of matching transliterations. The second model, which they call the generative transliteration model extends a word alignment hidden Markov model (W-HMM) (He 2007). In a W-HMM, the emission probability depends on the current character and the previous target character. By marginalizing over all possible alignments, they compute the probability of a target word given a source word as follows:

\[
P(T|S) = \sum_{A} \prod_{j=1}^{m} P(a_j|a_{j-1}, s_{a_j-1})P(t_j|s_{a_j}, t_{j-1})
\]  

(2.7)

where \(t_j\) (and respectively \(s_i\)) denote the \(j^{th}\) (and respectively \(i^{th}\) character in \(T\) (and respectively \(S\)) and \(A \equiv a_1^n\) is the hidden alignment between \(T\) and \(S\), and in which \(t_j\) is aligned to \(s_{a_j}\), for \(j = 1, \ldots, m\). The parameters of the W-HMM are estimated using the EM algorithm and they use \(\log P(T|S)\) for the transliteration similarity score.

Klementiev and Roth (2006) exploit two observations in mining transliteration pairs from multi-lingual news streams. The first observation is that NEs in one language associated with multi-lingual news streams tend to co-occur with their counterparts in another language for a given period of time. The second observation is that “NEs often contain or are entirely made up of words that are phonetically transliterated or have a common etymological origin across languages”. Klementiev and Roth (2006) introduce an algorithm called co-ranking which exploits the two observations simultaneously during the transliteration mining process. For the first observation, they use a Discrete Fourier Transform (Arfken 1985) based metric to compute the similarity of time distributions. For the second observation, they score NE similarity using a discriminative linear transliteration model. The transliteration model is iteratively trained using single word NE pairs. During training, for a given source NE \((S)\) in one language, the current model chooses a list of top-ranked
transliteration candidates $T$ in another language. The words in the candidate pair are partitioned into a set of substrings $su$ and $tu$ up to a particular length (including the empty string which they denote by $\_\_\_\_\_\_\_\_\_\_$). Couplings of the substrings $(su, tu)$ from the source and target language sets of words produce feature vectors which are used for training. Klementiev and Roth employ the perceptron (Rosenblatt 1958) algorithm which takes a variable number of features in its examples; and as the iterative algorithm observes more data, it discovers and makes use of more features. Time sequence scoring is then used to rank the list and subsequently choose the target candidate NE $T'$ that is best temporally aligned with $S$. A method called F-index (Hetland 2004) is used to implement the temporal similarity score function. The pairs of transliteration NEs and the best temporally aligned (thresholded) candidate NEs are utilized in a similar manner to iteratively train the transliteration model. The resulting pairs of source and target NEs are then evaluated for accuracy.

A recent shared task on transliteration mining (Kumaran et al. 2010b), used comparable Wikipedia article topics for five language pairs as source data for mining transliteration pairs. The comparable Wikipedia article topics are obtained with the aid of ‘inter-language links’ which are located on the same page as the source language text. A report about participating systems in the shared task shows the application of discriminative and generation-based methods for transliteration similarity estimation. Some approaches like those in Udupa et al. (2009) have been described above. Generation-based methods mostly included various forms of Hidden Markov Models (HMMs) and finite state automata (Darwish 2010, Noeman and Madkour 2010). Discriminative methods included some of the well known methods such as: support vector machines (SVMs) and a standard string kernel method (Jiampojamarn et al. 2010). The shared task report shows that some methods achieved good F-score values but on only one or a few datasets, and not for all datasets.

2.2.2 Discussion

The identification of candidate named entities can be a complex procedure depending on the requirements of the named entity recognition task and the language in use. As is suggested from some of the approaches reviewed above, some Asian language text presents more challenges for named entity recognition mainly because of the extra work required to identify words in a sentence as compared to text written using a Latin or Cyrillic alphabet. In Chinese text for example, the segmentation of a sentence is complicated by the lack of blanks and marks to indicate word boundaries. Consequently, the identification of candidate words is difficult with a major problem of segmentation ambiguities (Chen and Bai 1998). The review above also shows that some methods specify the transliteration similarity estimation problem in a similar
2.3 Transliteration generation

Unlike the transliteration mining task which can vary depending on the data source, the transliteration generation task (both forward and backward) is similar across different transliteration generation approaches. In forward transliteration, we want the transliteration system to automatically convert a source word into a target transliteration or a set of target transliterations when variations are expected. In the backward (reverse) direction, the aim is to find the original source word for a given target transliteration or transliterations. In both cases of transliteration generation, the core of the system is a trained model or set of models for character conversion. A general framework for transliteration generation is provided in (Karimi et al. 2011) specifying two main phases: training of the transliteration model(s) and generating transliterations using the trained model(s).

![Diagram of transliteration generation process]

Figure 2.1: Transliteration generation overview. Adapted from Karimi et al. (2011).

Karimi et al. (2011) identify different subtasks in each of the two main phases (Figure 2.1) in the transliteration generation process. Two common subtasks in the
training phase include: segmentation of source and target character strings in the training pairs, and determining associations between the source and target transliteration units after performing a training procedure or a manual specification of the transformation rules. In the transliteration generation phase, two common subtasks include: source word segmentation into transliteration units, and using the trained model(s) or transformation rules to map source transliteration units to target transliteration units by resolving different combinations of alignments and unit mappings. As illustrated in Figure 2.1, at the end of the transliteration generation phase the transliteration system is usually expected to suggest more than one target transliteration. Since there is always a dependence on the previous sub-task from the training phase to the transliteration generation phase, we will review some of the main transliteration generation approaches in their entirety. Automated transliteration generation approaches are usually categorized according to the type of transliteration units used; that is, whether they are phonetic or orthographic or a combination of both (Karimi et al. 2011, Oh et al. 2006). In our review, we follow a similar categorization.

2.3.1 Phonetic-based transliteration generation

The earliest reported attempts at automated transliteration generation involved the use of phonemes between the source and target languages. In Arbabi et al. (1994), the generation of romanized transliterations for Arabic names is as follows: The Arabic names are first vowelized automatically using a combination of an artificial neural network (ANN) and a knowledge-based system (KBS). The ANN is used to filter out words that would otherwise be vowelized inappropriately by the knowledge-based system; while the KBS uses linguistic vowelization rules. In the transliteration stage, the vowelized Arabic NEs can be converted into a standard, phonetic Latin representation using a parser or table. Generally, the Latin representation is broken down into a group of phonetic syllables which can be used to produce various spellings in languages that use the Latin alphabet.

A few years after Arbabi et al.’s (1994) work, Knight and Graehl (1997) used weighted finite-state automata for Japanese Katakana to English back transliteration. The transliteration process in Knight and Graehl (1997) follows a number of steps which are implemented as follows:

- English word (taken here as the source word S) sequences are generated using a distribution $P(S)$; then
- English pronunciations (SP) are produced from English words using $P(SP|S)$;
- the English pronunciations are converted into Japanese sounds TP using $P(TP|SP)$;
2.3 Transliteration generation

- Japanese sounds are converted into Japanese Katakana (T_k) using P(T_k|TP);
- and misspellings caused by Optical Character Recognition (OCR) (T_{OCR}) are modeled through P(T_{OCR}|T_k).

P(S) is implemented as a weighted finite state acceptor (WFSA), while the other conditional distributions are implemented as weighted finite state transducers (WFSTs). The aim is to find an English word S’ that maximizes the joint probability given a Katakana string (T_{OCR}) observed by OCR:

$$S' = \arg\max_S P(S) \cdot P(SP|S) \cdot P(TP|SP) \cdot P(T_k|TP) \cdot P(T_{OCR}|T_k)$$  (2.8)

Knight and Graehl (1997) use a general composition algorithm to integrate the different models and hence the computation in Equation 2.8. They then use Dijkstra’s shortest-path algorithm (Dijkstra 1959) for extracting S’. A unigram scoring method is used in constructing the WFSA for P(S); the WFST for P(SP|S) is based on the CMU pronunciation dictionary; the WFST for P(TP|SP) is learned automatically using an Estimation Maximization (EM) algorithm (Baum 1972) from a collection of English/Japanese sound sequences; the WFST for P(T_k|TP) is composed from two manually constructed WFSTs (the first WFST for merging long Japanese vowel sounds into new symbols while the second for mapping Japanese sounds to Katakana symbols); finally the WFST for P(T_{OCR}|T_k) is obtained using the EM algorithm applied on a collection of OCR’d text with the corresponding Japanese Katakana text.

Work related to Knight and Graehl (1997) then adapted and extended the phonetic-based approach while applying it to other language pairs. In Stalls and Knight (1998), weighted finite state automata are used in Arabic to English back-transliteration. Stalls and Knight (1998) use probability distributions similar to those in Knight and Graehl (1997) implementing the WFSA for P(S) and the WFST for P(SP|S) in exactly the same way. The modification in Stalls and Knight (1998) is that, instead of using conversions from English phonemes (SP) to Arabic phonemes (TP) and conversions from Arabic phonemes to Arabic orthography T, they use only one additional model for converting English phoneme sequences directly to Arabic orthography (P(T|SP)). The WFST for P(T|SP) is obtained using the EM algorithm on a manually built English-phoneme-to-Arabic-writing dictionary. In Al-Onaizan and Knight (2002), the phonetic-based approach in Stalls and Knight (1998) is adapted by using a finite state machine to filter out ill-formed English phonetic sequences instead of using position markers in the phoneme set during the Arabic to English back-transliteration process. Al-Onaizan and Knight (2002) also extended Stall and Knight’s (1998) phonetic based approach with the use of a spelling based model to deal with words that are not of English origin.
Apart from the back-transliteration work based on Knight and Graehl’s (1997) phonetic based approach, there were other parallel phonetic-based approaches that employed different techniques for transliteration generation in different language pairs:

Kawtrakul et al. (1998) performed Thai-to-English back-transliteration. In Kawtrakul et al. (1998), Thai loan words \((T)\) were first segmented into syllables and mapped to phonemes using some transcription rules. The phoneme sequences of the loan words were then compared to the phonetic sequence of a set of English words \((S)\) in a phonetic dictionary. The English word \((S’)\) with the most similar phonetic sequence was selected as the transliteration.

Jung et al. (2000) applied an extended Markov window method to build the model for English to Korean transliteration. In their transliteration process, Jung et al. (2000) first generate mappings between English and Korean phonemes. They use the pronunciation symbols for English words as defined in the Oxford computer-readable dictionary (Roger 1992) and construct English to Korean phonetic mapping tables that meet syllabification and alignment requirements for training the transliteration model. Their alignment process proceeds in two stages: consonant alignment obtained from a scan of English phonetic units and Korean notation; and vowel alignment which leads to a separation of corresponding vowel pairs based on the consonant alignment stage. In the transliteration generation stage, a probabilistic tagger is used to find the most likely Korean transliteration candidates given an English input \(S\) that has been syllabified. Given that \(S\) represents an English NE (where \(S_s = ss_1 ss_2 \ldots ss_n\) is its syllabification), and \(T = t_p t_p \ldots t_p\) a Korean word (where \(t_p\) is the \(i^{th}\) phonetic unit of \(T\)), Jung et al. (2000) aim at finding a Korean word \(T’\) such that the joint probability \(p(S, T) \approx p(S_s, T)\) is maximized.

\[
T’ = \arg\max_T P(S_s, T) = \arg\max_T P(T|S)P(S) = \arg\max_T P(T|S)P(S) \quad (2.9)
\]

where the translation model in Equation 2.9 is approximated based on the extended Markov window as follows:

\[
P(T|S) \cong \prod_i P(t_p_i|t_p_{i-1}, ss_{i-1}, ss_i, ss_{i+1}) \quad (2.10)
\]

Equation 2.10 is further expanded into more fragmented probability terms to deal with data sparseness when training. For the language model probability \(P(S_s)\) in Equation 2.9, a bi-gram language model is used:

\[
P(S_s) \cong \prod_i P(ss_i|ss_{i-1}) \quad (2.11)
\]
the transliteration model is then finally formulated as:

\[
T' = \arg\max_T P(S, T) \approx \arg\max_T \prod_i P(t_p_i|s_{s-i+1}) P(s_{s-i}|t_p_{i-1}) P(s_{s-i+1}|t_p_i, s_{s-i}) \frac{P(s_{s-i+1}|s_{s-i})}{P(s_{s-i}|t_p_i, s_{s-i})}
\]

(2.12)

Lee and Choi (1998) model English to Korean transliteration by using both phoneme transformations (pivot method corresponding to phonetic based transliteration), and only grapheme transformations (direct method). In their phoneme-based transliteration method, English graphemes are first converted to English phonemes; the English phonemes are then converted to Korean graphemes. Using S to denote an English word and T a Korean word, the aim is to find a Korean word T' that maximizes the conditional probability \(P(T|S)\). By applying Bayes’ rule, Lee and Choi specify their transliteration problem as:

\[
T' = \arg\max_T P(T|S) = \arg\max_T \frac{P(T)P(S|T)}{P(S)} \approx \arg\max_T P(T)P(S|T)
\]

(2.13)

The language model probability \(P(T)\) in Equation 2.13 is obtained by using a bi-gram conditional probability distribution on Korean pronunciation units (PUs):

\[
P(T) \approx \sum_t \prod_{i=1}^{N} P(t_p_i|t_p_{i-1})
\]

where N is the total number of segmentation of T, \(t_p_i\) is the \(i^{th}\) PU in a segmentation of T, and t is the total number of PUs in a segmentation. The translation model probability \(P(S|T)\) is also obtained using a bi-gram conditional probability distribution between English and Korean PUs:

\[
P(S|T) \approx \sum_{\delta} \prod_{i=1}^{t} P(s_{p_i}|t_p_i)\text{ where } \delta \text{ is the total number of alignments for } T \text{ and } S.
\]

Wan and Verspoor (1998) use five stages in English to Chinese transliteration. In the first stage they parse a complete English phrase through a dictionary in search of a standard translation, and if there is no standard translation, the phrase is broken into words, and each word is parsed through a dictionary. Words with no standard translations are selected for transliteration. In the second stage, each word that is selected for transliteration is divided into syllables, and in the third stage the transliteration process proceeds to find patterns within each syllable that are handled in appropriate ways for mapping to a particular Romanization standard (Pinyin in this case) in the fourth stage. In the last stage, the Pinyin representation of a word is mapped to Chinese Han characters using a Pinyin - Han character correspondence table.
Meng et al. (2001) use a number of modules based on the steps required to transform an English out of vocabulary word to Chinese. Meng et al. (2001) first detect Romanized Chinese names using a maximum-matching segmentation algorithm and then automatically acquire pronunciations for names other than Romanized Chinese names using either the PRONLEX pronunciation lexicon from LDC or an automatic letter-to-phoneme generation process (which is obtained through training on the PRONLEX lexicon by aligning words with the corresponding pronunciations in Viterbi-style for a one-to-one letter-to-phoneme mapping). Meng et al. then apply cross-lingual phonological rules to deal with some problems in English pronunciations. For phoneme alignments between English and Chinese, Meng et al. iteratively train a finite state transducer using a bi-lingual proper name list containing English names and their Chinese transliterations. Given an English phoneme sequence, they implement confusion matrices to produce alternative Chinese phoneme sequences prior to syllabification in a Chinese Romanization system (Pinyin in this case). The result is called a Chinese phoneme lattice. In the final stage, they search through the phoneme lattice to identify Chinese phonemes that constitute legitimate syllables. The resulting syllable graph is searched using the $\Lambda^*$ search algorithm to find the $N$ most probable syllable sequence using probabilities derived from the confusion matrix and a syllable bi-gram language model.

Karimi et al. (2011) review a number of additional phonetic-based transliteration generation approaches. To avoid a complete repetition of the review in Karimi et al. (2011), we provide a summary of some of these other phonetic based approaches while focusing on the transliteration models used.

Jeong et al. (1999) use a Hidden Markov Model (HMM) framework in Korean to English back-transliteration. The aim is to determine the most likely original English ($S'$) word that maximizes the conditional probability of an English word given a “foreign” word ($P(S|T)$). Using $S = s_1s_2...s_m$ to denote the English word, and $T = t_1t_2...t_n$ to denote the foreign word, Jeong et al. (1999) formulate the transliteration problem as follows:

$$S' = \arg\max_S P(S|T) = \arg\max_S P(s_1s_2...s_m|t_1t_2...t_n)$$

$$= \arg\max_S P(s_1s_2...s_m) \times P(t_1t_2...t_n|s_1s_2...s_m)$$

(2.14)

Using a phonetic representation ($sp_1sp_2...sp_p$) for the English string $S = s_1s_2...s_m$, Equation 2.14 leads to:

$$S' = \arg\max_S P(sp_1sp_2...sp_p) \times P(t_1t_2...t_n|sp_1sp_2...sp_p)$$

$$\cong \arg\max_S \prod_j P(sp_j|sp_{j-1}) \times P(t_j|sp_j)$$

(2.15)
2.3 Transliteration generation

The first term in Equation 2.15 \( P(s_{pj}|s_{pj-1}) \) corresponds to the transition probability between two states of an HMM while the second term \( P(t_j|s_{pj}) \) to the output probability in a given state. The computation in Equation 2.15 is effected using the Viterbi algorithm.

Oh and Choi (2002) use pronunciation and contextual rules for English to Korean transliteration. In Oh and Choi’s system, English pronunciation units are first aligned to corresponding phonemes, then the transliteration of English words to Korean words is achieved through a number of steps including: identification and processing of “complex word forms”; detection and processing of English words of Greek origin; chunking of aligned English pronunciation and phoneme data into two classes of pure English words and English words of Greek origin; and finally English phoneme to Korean conversion based on the use of English to Korean Standard Conversion Rule (EKSCR). Contextual rules are captured by observing errors from the use of EKSCR to a given number of randomly selected words which are not part of the test set.

Virga and Khudanpur (2003) apply the IBM source-channel model (Brown et al. 1993) in English to Chinese transliteration. The steps followed in their transliteration process include: 1) conversion of an English name into a phonemic representation using the Festival speech synthesis system; 2) translation of the English phoneme sequence into a sequence of generalized initials and finals (GIFs) which are the commonly used sub-syllabic units for expressing pronunciations of Chinese characters; 3) transformation of GIF sequences into Pinyin symbols without specifying tone; and 4) translation of the Pinyin sequence to Chinese character sequence. Steps 2 and 4 are accomplished using Brown et al.’s (1993) statistical translation model. For example, for step 2, the aim is to find the sequence of GIF symbols \( g' = g'_1 g'_2 ... g'_k \) that maximize the probability of a GIF sequence \( g = g_1 g_2 ... g_l \) given an English phoneme sequence \( SP = sp_1 sp_2 ... sp_j \):  

\[
g' = \arg\max_g P(g|sp) = \arg\max_g P(sp|g)P(g). \quad (2.16)
\]

Virga and Khudanpur (2003) estimate a trigram model for \( P(g) \) in Equation 2.16 using a CMU toolkit on a training portion of Chinese names. In the case of step 4, a trigram model with Good-Turing discounting and Katz back-off is estimated as the language model for the transformation of Pinyin sequences to Chinese characters.

Gao et al. (2005) used a direct model in English to Chinese transliteration as opposed to the indirect approach in the source channel model described above (Virga and Khudanpur 2003). Using TC = tc1 tc2 ... tcI to represent Chinese Pinyin sequences and SP = sp1 sp2 ... spj for the English phoneme sequence as above, Gao et al. (2005) reformulate the transliteration problem by rewriting Equation 2.16:

\[
TC' = \arg\max_{TC} P(TC|SP)P(TC) \quad (2.17)
\]
Gao et al. (2005) use an EM algorithm to find the Viterbi alignment per training pair for generating English phoneme to Chinese Pinyin mapping probabilities, which are subsequently encoded in a WFST. For the language model for $P(\text{TC})$, they train a syllable-based bi-gram model using the same instances of Chinese names that were used for building the WFST.

Most of the reviews in literature about phonetic-based machine transliteration hardly discuss any work of the kind in the last five years. Below we briefly review only two recent references where phonetic-based transliteration is used. In both references, an Indian language is involved in the phonetic based transliteration process. While phonetic-based transliteration is used in both references, orthographic-based transliteration is also reported and seems to be used more than the phonetic based approach.

Surana and Singh (2008) use pronunciations for foreign words in English to Indian language transliteration. They use the the CMU speech dictionary for lookup and for training a pronunciation estimation model. English words that are not of Indian origin are first converted to phonemic representation and the English phonemes are then mapped to Indian language letters. For English words that are of Indian origin, they simply segment the English word and convert the Latin segments into Indian language segments for generating a corresponding Indian transliteration.

Das et al. (2010) use a phonetic based transliteration approach to handle valid English dictionary words. They use a standard machine learning sequence labeler conditional random field to map English phonemes to Indian language transliteration units.

### 2.3.2 Orthographic-based transliteration generation

Going by the common definition for transliteration generation, the transliteration process is expected to involve a phonetic mapping from one language to another. However, transliteration work as reported in recent literature suggests that, orthographic-only based methods result in transliteration generation quality which is comparable to that for phoneme-based methods, and sometimes even much better (Li et al. 2004). An orthographic-based approach eliminates a number of intermediate phonetic representation and transformation steps that require extra work and time. The elimination of the intermediate phonetic steps implies that any flaws that may be associated with the intermediate steps will be avoided. Recent work has mostly favored the use of orthographic-based transliteration generation, mainly applying techniques associated with the machine learning paradigm. Most of the earlier orthographic-based methods have been reviewed well in recent reviews on machine transliteration (Oh et al. 2006, Chinnakotla et al. 2010, Karimi et al. 2011). The orthographic-based
approach constitutes various types of approaches with two notable categorizations as: generative and discriminative. Although we discuss some of the transliteration generation methods under the two common categorizations of generative-based and discriminative-based approaches, we also separate the discussion for recent approaches that are motivated by work from a related domain (especially machine translation (Matthews 2007, Finch and Sumita 2008) in this case), and those that introduce new concepts such as semantic transliteration (Li et al. 2007).

a) Generative and Rule-based approaches

The earliest cited work on orthographic-based transliteration generation suggests a common usage of a machine translation inspired framework of the source channel model (Lee and Choi 1998, Jeong et al. 1999, Kim et al. 1999) and related methods such as the joint source channel model (Li et al. 2004, Zhang et al. 2004) and modified joint source channel model (Ekbal et al. 2006). Other earlier transliteration generation modeling methods include: decision trees (Kang and Choi 2000), transliteration networks (Kang and Kim 2000, Goto et al. 2003), and n-gram models (Abduljaleel and Larkey 2003). All of these are generation-based approaches. Relatively recent approaches include: Weighted Finite State Transducers (WFSTs) (Lindén 2006), hand crafted transliteration rules (Malik 2006), consonant-vowel-based methods (Karimi et al. 2006, Karimi et al. 2007), and substring-based transduction (Sherif and Kondrak 2007b). In the following we point out the main techniques that are employed starting with earlier approaches to recent approaches.

Kang and Choi (2000) used decision trees to generate Korean strings given English words. They use an extended version of Covington’s (1996) alignment algorithm to determine alignments for training the decision trees. In the extended version of Covington’s alignment algorithm, they introduce a binding operation to deal with ‘null mappings’. A depth-first search algorithm is used to prune away fruitless branches when estimating the alignments. To learn decision trees using the alignments, ID3 like algorithms (Quinlan 1986) are used. During transliteration, each English letter in a given English word is mapped to Korean characters using the corresponding decision trees; the Korean characters are then concatenated to produce the final Korean transliteration.

Goto et al. (2003) use several models based on a lattice of conversion units between English and Japanese Katakana characters. During transliteration, Goto et al.’s method follows three approaches: the computation of the likelihood of a particular choice of generating English conversion units through letter chunking for a given English word; the use of English and Japanese contextual information simultaneously to compute the plausibility of conversion from each English conversion unit to various Japanese conversion candidate units using a single appropriate probability
A review on machine transliteration

model; the use of several probability models based on the maximum entropy method while modeling different kinds of information.

Abduljaleel and Larkey (2003) use an n-gram transliteration model for English to Arabic transliteration. The model is a set of conditional probability distributions over Arabic characters, conditioned on English unigrams and selected n-grams. They use proper name lists to train the n-gram model using GIZA++ (a statistical alignment tool). During transliteration, an English word \( S \) is first segmented according to the n-gram inventory, and for each segment, all possible Arabic transliterations \( T \) are generated. The equation for scoring each word is given as follows:

\[
P(T|S, T \in \text{Ar}) = P(T|S) \times P(T \in \text{Ar})
\]

where \( P(T \in \text{Ar}) \) is the probability that the Arabic word \( T \) conforms to the spelling patterns of Arabic names, and is computed using a letter bigram model of general Arabic as the product of the probabilities of each letter bigram in \( T \).

Li et al. (2004) and Zhang et al. (2004) use a joint source channel model to capture the simultaneous generation of source and target words. A joint probability model is estimated and is marginalized to yield conditional probability models for both forward transliteration and back-transliteration. Given an alignment \( \delta \) with transliteration unit correspondences \( \langle s, t \rangle_k \) for an English string \( S \) and a Chinese string \( T \), Li et al. formulate their transliteration problem as follows:

\[
T' = \arg\max_{T, \delta} P(S, T, \delta) \text{ for English to Chinese transliteration, and}
\]

\[
S' = \arg\max_{S, \delta} P(S, T, \delta) \text{ for Chinese to English transliteration.}
\]

Li et al. (2004) then use an n-gram transliteration model to capture the conditional probability or transliteration probability of a transliteration unit correspondence \( \langle s, t \rangle_k \) depending on its immediate n predecessors. The following Equation is used to compute the joint probability of a pair of English (\( S \)) and Chinese (\( T \)) words:

\[
P(S,T) = P(S, T, \delta) = \prod_{k=1}^{K} P(\langle s, t \rangle_k|\langle s, t \rangle_{k-n+1}^{k-1})
\]

Lindén (2006) aligns source and target words using the minimum simple edit distance. From the alignments, Lindén determines the frequency of each edit operation in context for at most four letters in the source word including the letter \( s_i \) which is aligned with \( t_i \) in the target word \( T \). Given that the context \( s_{i-1}, s_i, s_{i+1}, s_{i+2} \) in the source word is represented by \( s_{i4} \), Lindén formulates the transliteration problem as follows:

\[
P(T|S) = \prod_{i=1\ldots \max(|T|,|S|)} P(t_i|s_{i4})
\]
2.3 Transliteration generation

\( P(t_i|s_{i4}) \) is estimated with counts of the transformations \( t_i|s_{i4} \) divided by the count of the context \( s_{i4} \). When the context rarely occurs, an offline back-off model is used for smoothing \( P(t_i|s_{i4}) \). Lindén then uses a cascade of weighted finite state transducers to implement the transliteration process.

Malik (Malik 2006) employs a completely rule-based approach for transliteration from Shahmukhi to Gurmukhi. During transliteration, each Shahmukhi token is parsed into its constituent characters. Characters that bear a dependency are transliterated using ‘dependency rules’ while characters that do not bear a dependency are transliterated by character mapping. Malik (Malik 2006) specifies a number of tables that encode the different types of rules.

Karimi et al. (2007) use an alignment approach comprised of two steps: the first step uses consonant and vowel properties of a word’s characters, and the second uses a frequency-based search for valid alignments. In the first step, consonant-vowel sequences (qs and qt) for a pair of words (S and T) in a training corpus is generated and if the sequences match, consonant clusters and vowel sequences are added to an alignment set. If qs and qt do not match, the second step is used where a search for alignments proceeds from left to right while examining one of four possible options for transliteration: single character to single character \((s_i, t_j, r)\), digraph to single character \((s_i, s_{i+1}, t_j, r)\), single character to digraph \((s_i, t_j, t_{j+1}, r)\), and single character to empty string \((s_i, \epsilon, r)\). For transliteration generation, Karimi et al. (2007) propose the use of a collapsed consonant and vowel method called (CV-MODEL3) as an extension of two previous models (CV-MODEL1 and CV-MODEL2). The source word is segmented and a probability is computed for each generated word \((T)\) as follows:

\[
P(T|S) = \prod_{k=1}^{[K]} P(\hat{T}_k|\hat{S}_k),
\]

where \([K]\) is the number of distinct source segments and \(P(\hat{T}_k|\hat{S}_k)\) is the probability of the \(\hat{S}_k \rightarrow \hat{T}_k\) transformation rule. Karimi et al then apply a tree structure following Dijkstra’s \(\alpha\)-shortest path, to generate the \(\alpha\) highest scoring (most probable) transliterations, ranked based on their probabilities.

b) Semantic transliteration

Li et al. (2007) introduce into the transliteration model semantic information with regard to language of origin and the gender associated with a name. The aim is still to determine the optimum target name \(T’\) which yields the highest posterior
probability given the source name $S$:

$$T' = \arg\max_{T \in \tau_S} P(T|S)$$

(2.21)

where $\tau_S$ is the set of all possible transliterations for the source name $S$. To incorporate language of origin ($L$) and gender information ($G$) in the transliteration, Equation 2.21 is re-written as:

$$P(T|S) = \sum_{L \in \mathcal{L}, G \in \mathcal{G}} P(T, L, G|S) = \sum_{L \in \mathcal{L}, G \in \mathcal{G}} P(T|S, L, G)P(L, G|S)$$

(2.22)

where $P(T|S, L, G)$ is the transliteration probability from source $S$ to target $T$, given the language of origin $L$ and gender $G$ labels. $\mathcal{L}$ and $\mathcal{G}$ denote the sets of languages and gender respectively. Given an alignment between $S$ and $T$, $P(T|S, L, G)$ is estimated using a bigram language model. The mappings between source and target characters for computing $P(T|S, L, G)$ are obtained from alignments resulting from the application of the EM algorithm on training data. Information concerning gender and language of origin is incorporated in Equation 2.22 by rewriting $P(L, G|S)$ as $P(L, G|S) = P(G|L, S)P(L|S)$. Using $L_S$ to denote the language of $S$, $P(L|S)$ can be obtained as:

$$P(L|S) = \begin{cases} 1 & L = L_S \\ 0 & L \neq L_S \end{cases}$$

and using $G_S$ to denote the gender for $S$, $p(G|L, S)$ is obtained as:

$$P(G|L, S) = \begin{cases} 1 & G = G_S \\ 0 & G \neq G_S \end{cases}$$

In the case where semantic information is not available, Li et al. learn the semantic information from the names themselves.

c) Transliteration using statistical machine translation methods

While a number of approaches aim to develop transliteration-specific methods, the adaptation of statistical machine translation (SMT) methods to transliteration generation has become popular as a valuable alternative. The SMT methods that have been used for transliteration generation range from the earlier popular IBM models (Brown et al. 1993) to a currently more popular state-of-the-art phrase-based SMT approach (Koehn et al. 2003). The application of the SMT methods is simply modified to reflect the properties of a transliteration process. That is, the task is first viewed as a character translation problem rather than a word (or phrase-based translation)
2.3 Transliteration generation

problem. In section 2.3.1 on phonetic based methods for transliteration generation, we have already seen the application of the IBM models (Virga and Khudanpur 2003). Orthographic-based methods that use the IBM models, utilize them in a manner similar to that presented for the phonetic based approaches. A detailed description about the adaptation of the phrase-based SMT approach to transliteration can be found in Matthews (2007) and Finch and Sumita (2008).

d) Discriminative machine transliteration

Zelenko and Aone (2006) propose two discriminative methods for transliteration. Using an existing transliteration dictionary $D$ (a set of name pairs $(S,T)$), Zelenko and Aone learn a function that directly maps a name $S$ from one language into a name $T$ in another language. The main difference in their work is the omission of the alignment step and any probabilistic computations such as $P(T|S)$, $P(S,T)$ that depend on alignments. Their discriminative methods correspond to local and global modeling paradigms: in the local paradigm, Zelenko and Aone learn linear classifiers that predict a letter $t_i$ from the previously predicted letters $s_1...s_{i-1}$ and the original name $S$. In the global paradigm, Zelenko and Aone learn a function $W$ for mapping a pair $(S,T)$ into a score $W(S,T) \in \mathbb{R}$.

Klementiev and Roth (2006) train a linear model to decide whether a target word $(T)$ from a set of candidate words is a transliteration of a source word $(S)$. $T$ and $S$ are partitioned into a set of substrings $S_s$ and $T_s$ up to a particular length (including the empty string). Klementiev and Roth use the same transliteration model described in the previous section on Transliteration Mining to convert input strings to target strings.

e) NEWS 2009 and NEWS 2010 shared tasks on transliteration generation

The transliteration generation task has also led to the organization of two transliteration generation shared tasks (Li et al. 2009, Li et al. 2010) to evaluate state-of-the-art transliteration generation methods using the same standard corpora. The reports from the two shared tasks (Li et al. 2009, Li et al. 2010) show the use of both generation-based and discriminative-based techniques in transliteration generation modeling.

The 2009 NEWS shared task report (Li et al. 2009) identified two transliteration generation modeling approaches that were applied by many of the participating systems: phrase-based statistical machine transliteration (which originates from statistical machine translation work as described above) and Conditional Random Fields (CRFs) (Lafferty et al. 2001). The most successful approaches in the first shared task, however, combine several models (CRFs, Maximum Entropy Models, Margin Infused
Relaxed algorithm) by re-ranking the transliteration generation outputs from each model (Oh et al. 2009). A discriminative sequence prediction model (Jiampojamarn et al. 2009) referred to as DirectL is also reported to perform well.

The 2010 NEWS shared task report (Li et al. 2010) shows reduced participation, but for the systems that participated, approaches that are similar to those applied in the 2009 shared task are used including phrase-based machine transliteration and CRFs. The phrase-based approach is used for transliteration on various language pairs (Finch and Sumita 2010, Song et al. 2010) while CRFs are used by one of the seven participating systems (Das et al. 2010). Jiampojamarn et al. (2010) extend their DirectL approach above resulting in relatively better transliteration generation quality for this shared task. Most of the participating approaches also combine different models via re-ranking of the outputs (Das et al. 2010, Finch and Sumita 2010, Song et al. 2010) to improve transliteration generation quality. All methods are said to be orthographic-based except for some cases where a Romanization system is used before applying a transliteration generation system. However, in almost all results associated with the twelve language pairs for the 2010 shared task, only one participating system achieves just over 50% transliteration generation word accuracy on the English-Korean language pair.

2.3.3 Discussion

Approaches for automated transliteration generation can be divided into two main classes: those that use phonetic information in the transliteration process and those that use only the orthographic representation. Transliteration generation literature seems to put the two classes into different timelines. The phonetic-based approaches were mainly utilized in earlier when research on the task had just started. Later and more recently, orthographic-based approaches seem to be more preferred. However, there are also techniques that combine the use of both phonetic and orthographic-only information. Many approaches incorporate linguistic information to help improve transliteration generation quality. Recently, there are attempts to incorporate different types of information such as semantic information other than using only the orthographic representations (Li et al. 2007). Reports for the recent shared tasks on transliteration generation (Li et al. 2009, Li et al. 2010) also show that a combined application of methods leads to improved transliteration generation quality with respect to individual methods. This is achieved via a comparative analysis or an integration of the outputs from individual systems to get the final result. Results associated with the 2010 shared task suggest that there is need to improve transliteration generation quality as almost none of the state-of-the-art systems achieved over 50% transliteration generation accuracy. Evaluation of the state-of-the-art systems
2.4 Conclusion

at the time of writing this thesis is ongoing.

2.4 Conclusion

Literature on automated transliteration mining and generation tasks suggests generic constituent phases for which various methods have been proposed to model transliteration related tasks. For transliteration generation, many approaches follow an overall two step procedure of first specifying or training transliteration models given correct transliteration pairs, and then later applying the models to propose target candidate words given a source language word. It is usually the case that the size of training data affects the representational quality and consequently the performance of the transliteration models.

Recent shared tasks on transliteration mining (Kumaran et al. 2010b) and generation (Li et al. 2009, Li et al. 2010) provide us with some baseline for making general conclusions about the current state of research. Literature also shows that most of the approaches actually do employ similar techniques with very few modifications. For example, the reports for all the previous shared tasks on transliteration generation show that a large percentage of the systems used either a phrase-based statistical machine translation approach or conditional random fields. While some of these approaches achieved reasonable performances on some language pairs, recent shared task results show that they are still far from achieving high transliteration generation and mining quality for many language pairs.

Literature shows that the use of the setups for transliteration mining and generation have only been for cases where the source and target language use different writing systems. In this thesis, we propose the application of the traditional transliteration generation setups for cases where source and target use the same writing system.

Finally, while literature shows the application of the classic HMMs in both transliteration mining and generation, there seems to be no report concerning the evaluation of other DBN approaches like the ones we have proposed in the thesis for the two tasks.
3.1 Introduction

Dynamic Bayesian networks (DBNs) are a class of temporal probabilistic graphical models (PGMs) that have found successful application in many domains. This is attributed to the more general representations that DBNs allow, leading to very large model spaces and the use of generic algorithms for inference and learning. The DBN framework already generalizes a variety of methods including some of the most common and successful methods in Natural Language Processing (NLP) such as Hidden Markov Models (HMMs). The inference and learning algorithms used in these methods can also be viewed as instantiations of some of the standard DBN algorithms; for example, the forward-backward algorithm used for inference and learning in HMMs can be considered a special type of the message-passing algorithm used for inference in Bayesian Networks.

As temporal probabilistic graphical models, DBNs are used to model not only sequential data (linguistic or biological) where we are doubtful about the generating mechanism but also to model time series data that is generated by some causal process. A variety of DBN modeling methods have been proposed in the literature, most of which are easily adaptable to a variety of tasks.

In this chapter, we present the concepts underlying the framework of Dynamic Bayesian Networks with respect to three aspects: the specification of the models, inference, and learning methods. But first, we give an introduction to Bayesian networks from which DBNs are an extension.
3.2 Bayesian networks

3.2.1 Representation

Bayesian networks provide a means of expressing a joint probability distribution over a set of inter-related random variables. They are specified by way of a graphical modeling language where nodes are used to represent random variables and edges are used to represent dependencies between the random variables for some system domain. The use of a graphical language to communicate representative models carries with it a number of advantages. First of all, a graphical representation provides an accurate reflection of our understanding of the domain we are modeling and facilitates the effective construction of the models. Secondly, a graphical representation allows the distributions defined by a given model to be used effectively for inference where there is need to answer different types of queries with respect to the problem domain.

Bayesian networks are specifically directed acyclic graphs in which edges specify conditional dependencies or independencies. The graphical representation of a network also specifies the requirements for the quantitative part of the model which comprises of a set of probability distribution functions for each random variable. Formally, a Bayesian network is defined to have the following (Jensen and Nielsen 2007, Koller and Friedman 2009):

- A set of random variables $X_1, ..., X_n$ represented by nodes and a set of directed edges between the random variables.

- Each variable has a finite set of mutually exclusive states.

- The variables together with the directed edges form a directed acyclic graph (DAG).

- Each variable $X_i$ is conditionally dependent on only its parents ($\text{Pa}_{X_i}$) and its children.

- To each variable $X_i$ with parents $\text{Pa}_{X_i}$, a conditional probability table (CPT) for $P(X_i|\text{Pa}_{X_i})$ is attached.

As suggested by the definition above, a number of stages are involved in constructing a Bayesian network model to represent a given domain including the following (Koller and Friedman 2009):

1. **Data Collection**: Collect data relevant to the domain of interest.
2. **Structure Learning**: Construct the graphical structure of the network.
3. **Parameter Estimation**: Estimate the parameters of the CPTs.
4. **Inference**: Use the network to answer queries and make predictions.
3.2 Bayesian networks

a) Specifying variables

The variables in a Bayesian network represent entities that are relevant to the domain we are modeling. These entities and their related attributes can be described in various ways, but it is important that the variables that we specify do precisely represent the domain. In doing so, we partly ensure that a conclusion resulting from the use of the network is accurate with respect to a given set of observations in the domain. In addition to specifying variables that precisely represent the domain, we also have to specify each variable’s domain of values which should be representative enough so as to ensure true conditional independence assumptions with respect to the Bayesian network model. With respect to the problem, different types of variables can be specified including observable and/or hidden variables. Observable variables are variables that we can directly measure while hidden variables are variables that we can only infer from the observable variables in the model.

b) Specifying the Bayesian network structure

Using the graphical modeling approach, the specification of a Bayesian network structure is not straightforward because there can be many structures that reflect the same set of independencies in the domain we are modeling. Koller and Friedman (2009) suggest an approach that should be successful for specifying a Bayesian network to represent the domain of interest. This approach according to Koller and Friedman is to specify a structure that in the most part reflects the causal order and dependencies of the variables in the domain, so that between two variables, we use the notion ‘child variable’ to represent the effect and the notion parent variable to represent the cause. Koller and Friedman (2009) also point out a commonly used approach of using a backward construction process in the specification of a Bayesian network structure. The process is called backward because when constructing the Bayesian network, we start with a child variable which represents an effect and move on to determine factors (causes) that are associated with the effect which we add to the network as parent variables having edges to the child variable. Koller and Friedman (2009) also emphasize the inevitability of approximations where it is possible to represent many weak dependencies which can easily result in a very complex model that is infeasible to use.

c) Specifying probabilities

This is a very important task in completing the specification of a Bayesian network model since the queries that we would like to answer about the domain modeled by the network rely on the probabilities encoded by the Bayesian network. Koller and
Friedman (2009) point out a number of errors that can have a significant effect on the conclusions that result from using a Bayesian network. Examples of errors from Koller and Friedmann (2009) that we need to address when specifies probabilities for different variables in the network include: zero probabilities which need to be avoided in appropriate ways; the presence of small differences in probabilities that could imply large differences in conclusions; and the insensitivity of the model to differences in relative probabilities, for example we would require that the Bayesian network model encode correctly that the probability of the relationship between an English \( l \) and a Russian \( н \) is greater than the relationship between an English \( l \) and the Russian \( ж \).

3.2.2 A transliteration example

We would like to represent the character relationships between writing systems used by some source and target languages (for example English and Russian respectively). We can use a random variable \( s_i \) to represent a character in the source language writing system, and a random variable \( t_j \) to represent a character in the target language writing system. \( i \) and \( j \) are used to map to the \( i^{th} \) (respectively \( j^{th} \)) character in the source (respectively target) set of characters.

![Bayesian network graph](image)

**Figure 3.1:** A Bayesian network graph for relating characters between writing systems. \( s_i \) and \( t_j \) are random variables representing unique characters in the source (resp. target) writing system. \( V_s \) and \( V_t \) equal the total number of characters in the source (resp. target) writing system.

We can represent the the relationship between the source and target language characters by making \( t_j \) ‘depend’ on \( s_i \) as shown in Figure 3.1. To complete the definition of the Bayesian network for this example, we need local probability models to represent the nature of the dependence of \( t_j \) on \( s_i \). According to Figure 3.1, we need a probability model \( P(s_i) \) to represent the distribution of the different characters in the source writing system. The distribution over the target random variable \( t_j \).
3.2 Bayesian networks

is a conditional distribution which we denote here as $P(t_j|s_i)$. $P(t_j|s_i)$ means that for each assignment of values for the source random variable $s_i$, there is a different distribution for $t_j$. For example, given that the variable $s_i$ is assigned the English character ‘a’, there is a probability distribution for each Russian character $t_j$ that appears as a corresponding character in a transliteration where ‘a’ appears.

### 3.2.3 Bayesian networks – inference

The main importance of a Bayesian network model is its use for answering queries related to the problem domain that it models. The reasoning process that is followed in using a Bayesian network model for answering a given query is referred to as inference. In Bayesian networks, the general inference task is to compute the posterior distribution over a subset of variables (query variables) given the values of some evidence variables. The computation can involve hidden variables which are neither query nor evidence variables. In the example of Figure 3.1 where the network models the similarity between characters in different writing systems, a query we may want to answer is what probability should be associated with assigning the variable $t_j$, some target character, given that $s_i$ is assigned to some source character, for example $P(t_i = ш|s_i = а)$ (in the case of a comparison of Latin ‘a’ and Cyrillic ‘ш’ characters). The value for this probability based on the network of Figure 3.1 is easy to obtain as we only need to look it up from the conditional probability distribution for the variable $t_i$. If the question is inverted, that is $P(s_i = а|t_i = ш)$, we can still easily arrive to an answer by doing inference by enumeration where we utilize Bayes theorem:

$$P(s_i = а|t_i = ш) = \frac{P(s_i = а, t_i = ш)}{P(t = ш)} = \frac{P(s_i = а) \times P(t_i = ш|s_i = а)}{\sum_{s_i} P(t_i = ш, s_i)} \tag{3.1}$$

Bayesian network inference done in this way leads to an exact answer and is therefore called exact inference. Generally, exact inference requires a summation over a joint distribution in which we marginalize out irrelevant variables. The direct computation of probabilities as illustrated in Equation 3.1 is only possible for a small network. Several exact inference algorithms have been proposed to help increase efficiency over the direct approach while still being applicable for some complex representations. The following are some of the exact inference algorithms for Bayesian networks (Darwiche 2008): inference by variable elimination; inference by tree clustering; inference by conditioning; inference by reduction to logic.

As the size of the network increases with respect to the number of random variables and connections between them, we may experience an exponential blow up of the joint probability distribution represented by the model. All the exact inference
3. Dynamic Bayesian Networks

methods mentioned above are sensitive to this complexity. Approximate inference algorithms are insensitive to this complexity and can be quite efficient regardless of the network topology.

The Bayesian network in Figure 3.1 can only be used from a static point of view. That is, the joint distribution over the variables \( s_i \) and \( t_i \) of the network is fixed although there are different values for \( P(t_i|s_i) \). In a temporal setting such as the transliteration similarity task where we want to represent a distribution over a sequence of characters, the Bayesian network of Figure 3.1 cannot be used. In order to deal with problems where there is need to define distributions over more complex inter-relationships such as those in a temporal setting, template-based approaches have been proposed (Koller and Friedman 2009). In this thesis we use the language of dynamic Bayesian networks as a template-based approach for modeling transliteration similarity.

3.3 Dynamic Bayesian Networks

The use of Bayesian networks as described in the previous section is only limited to representing a domain statically. Dynamic Bayesian networks have been developed to extend Bayesian networks to enable the representation and analysis of systems that change over time. The approach builds upon the framework of Bayesian networks but where random variables in the Bayesian network relate to time. DBNs are usually discussed under three main aspects in the literature (Koller and Friedman 2009, Murphy 2002): representation, inference, and learning. We follow the same outline while using most of the notation as in Murphy (2002).

3.3.1 DBNs – representation

The possibility to have random variables relate to time in a Bayesian network enables DBNs to represent probability distributions over a sequence of random variables comprising of observations that are related to an underlying sequence of hidden states. A DBN model is formally defined as a pair \( \langle B_0, B_\rightarrow \rangle \) where \( B_0 \) is a Bayesian network over an initial distribution over states \( P(Z_1^{1:N}) \), and \( B_\rightarrow \) is a two-slice Temporal Bayes net (2-TBN) (Murphy 2002). Just as in Bayesian networks, the structure of a DBN is a directed acyclic graph (DAG) where each node represents a domain variable of interest at some time instant, and each directed arc represents the dependency between the two nodes it connects. A hidden state is represented in terms of a set of \( N_h \) random variables, \( S_t \) and the observation is also represented in terms of a set of \( N_o \) random variables, \( O_t \). The transition and observation models of a DBN are defined as a product of the conditional probability distributions (CPDs) in the
2-TBN (Murphy 2002):

\[ P(Z_t | Z_{t-1}) = \prod_{i=1}^{N} P(Z_t^{(i)} | Pa(Z_t^{(i)})) \]  

where \( Z_t^{(i)} \) is the \( i \)th node at time \( t \) (which may be hidden or observed; and \( N = N_h + N_o \)), and \( Pa(Z_t^{(i)}) \) are the parents of \( Z_t^{(i)} \), which may be in the same or previous time-slice (assuming a first-order Markov model). A given time \( t \) is associated with a number of states each of which is associated in turn with a number of parents \( Pa(Z_t^{(i)}) \) which may influence \( Z_t^{(i)} \). The product of these characterizes the state at time \( t \), \( Z_t \).

For a given DBN, we can generally assume that the parameters associated with nodes and dependencies among nodes are time invariant; and in particular, that the dependency parameters between two nodes across two time slices remain unchanged with time. Hence, as Murphy (2002) puts it, we can define a DBN over an observation sequence of length \( T \) by “unrolling” the 2-TBN until we have \( T \) time-slices. The joint distribution for the sequence of length \( T \) can then be obtained by multiplying together all of the CPDs (Murphy 2002):

\[ P(Z_{1:T}^{(1:N)}) = \prod_{i=1}^{N} P_{B_{0}}(Z_1^{(i)} | Pa(Z_1^{(i)})) \times \prod_{t=2}^{T} \prod_{i=1}^{N} P_{B_{t-1}}(Z_t^{(i)} | Pa(Z_t^{(i)})) \]  

Figure 3.2 is a graphical representation of a 2-TBN for a Hidden Markov model (HMM) (a) and when unrolled for \( T=4 \) slices. HMMs are the simplest of DBN models as they represent the hidden state using only one random variable.

![Figure 3.2: (a) A 2-TBN for a Hidden Markov Model and (b) An unrolled network of four time-slices for the HMM. We follow the common convention of representing observed variables as shaded nodes and hidden variables as clear nodes.](image-url)
3. Dynamic Bayesian Networks

3.3.2 DBNs – Inference

In DBNs, the problem of inference is generally represented as the problem of finding the probability of hidden variables in a time-slice given a set of consecutive observations. The most important literature on DBN inference (Murphy 2002, Koller and Friedman 2009, Mihajlovic and Petkovic 2001) identifies four common DBN inference tasks: filtering, prediction, smoothing, and decoding. Let $S_t$ and $O_t$ denote the state and observation at time $t$:

In filtering, at time $t$, inference is done to keep track of $P(S_t|O_{1:t})$, that is, we estimate the ‘belief’ state given all of the observations (evidence) obtained so far.

In prediction, given observations $O_{1:t}$, inference is done to predict the distribution over some subset of variables at time $t’ > t$.

Smoothing involves estimating a state of the past, given all the evidence up to the current time in some longer trajectory: $P(S_{t-l}|O_{1:t})$, where $l : 0 \leq l \leq t$ can vary with different time ranges. Smoothing in this case is aimed at incorporating future evidence to help reduce temporary fluctuations in the belief state which can lead to temporary “misconceptions” in the belief state (Koller and Friedman 2009).

In decoding, the aim is to find the most likely sequence of hidden states given the observations: $S_{1:t}^* = \arg\max_{S_{1:t}} P(S_{1:t}|O_{1:t})$

3.3.3 DBNs – Learning

There are two types of learning that are associated with DBNs: DBN structure learning and DBN parameter estimation. In DBN structure learning, the task is to extract a DBN structure as well as its parameters given training data. In DBN parameter estimation, we assume that the DBN structure is known and the learning task is to determine the parameters that define the conditional probability distributions of the attributes.

a) DBNs – structure learning

Since a DBN can be represented by two networks $B_0$ and $B_→$, learning the structure of a DBN reduces to learning the structure of $B_0$ and $B_→$. Boyen et al. (1999) provide a detailed introduction to DBN structure learning. Here, we briefly point out the most relevant parts of their discussion.

If we have complete data (that is the training sequence $D$ is fully observable), the learning task is to find the network $B→$ that “best matches” $D$ (Friedman et al. 1998). The notion of best match is defined using a scoring function and the term of interest is the log-likelihood function, defined as $L(B : D) = \log P(D|B)$. The log-likelihood function measures how likely the data is given a candidate model $B$. The
**3.3 Dynamic Bayesian Networks**

A **log-likelihood** function relies on **sufficient statistics** that summarize the frequencies of the relevant events in the data. A scoring function that utilizes the log-likelihood function can be defined and the goal then is to find a network that maximizes the score.

If we have incomplete data (that is, the training sequence $D$ is partially observable), then we no longer know the exact counts in the data. As one of the earliest attempts at DBN structure learning, Friedman (1997) extends the traditional Expectation Maximization (EM) algorithm to a Structural EM (SEM) algorithm.

**b) DBNs – parameter learning**

In DBN parameter estimation, we start with some apriori knowledge about the DBN model which is represented in the form of a prior probability distribution over model parameters. The knowledge is updated using data to obtain a posterior probability distribution over models and parameters. More formally, assuming a prior distribution over the parameters for a given DBN model structure $P(\theta|B)$, a data set $D$ is used to compute a posterior distribution over the parameters.

$$P(\theta|B, D) = \frac{P(D|\theta, B)P(\theta|B)}{P(D|B)}.$$  \hfill (3.4)

**c) The Expectation Maximization algorithm and its generalization**

A common approach that is used to estimate the parameters given that the model has hidden variables is Maximum Likelihood Estimation (MLE) using an expectation maximization (EM) algorithm.

Although the EM algorithm is well covered in the Literature, it is important to review it here since it plays the most important role in the training of the DBN models that we apply to transliteration similarity estimation. Our review of the EM algorithm is completely based on an unpublished note by Stuart Russell and on a tutorial by Borman (2004) which also builds upon Stuart Russell’s note. We also use similar notation as in Russell’s note and in Borman’s (2004) tutorial.

The expectation maximization algorithm iterates between two steps: the expectation step where we compute values of hidden variables and observation variables given training data and model parameters; and a maximization step, where model parameters are estimated that maximize the likelihood of training data. We aim at finding $\theta$ such that $P(D|\theta)$ is maximal. The maximization step of the EM algorithm can be achieved by introducing a **log likelihood function** of the parameters $\theta$ given the data $D$:

$$\mathcal{L}(\theta) = \ln P(D|\theta)$$  \hfill (3.5)
Using $\theta_n$ to represent the current estimate for the parameters after $n$ iterations, we wish to find an estimate of the parameters in the next iteration ($\theta_{n+1}$) such that the difference $\mathcal{L}(\theta_{n+1}) - \mathcal{L}(\theta_n)$ is maximized. In the presence of a set of hidden variables, denoted by $Z$, $P(D|\theta_{n+1})$ can be written as:

$$P(D|\theta_{n+1}) = \sum_z P(D|z, \theta_{n+1})P(z|\theta_{n+1})$$

(3.6)

where $z$ refers to the values of $Z$. The difference that we wish to maximize can then be written as:

$$\mathcal{L}(\theta_{n+1}) - \mathcal{L}(\theta_n) = \ln \left( \sum_z P(D|z, \theta_{n+1})P(z|\theta_{n+1}) \right) - \ln P(D|\theta_n).$$

(3.7)

As Equation 3.7 involves the logarithm of a sum, Jensen’s inequality (Jensen 1906) which states the following can be used:

$$\ln \sum_{i=1}^{n} \lambda_i x_i \geq \sum_{i=1}^{n} \lambda_i \ln(x_i) \quad \text{where } \lambda_i \geq 0 \text{ are constants such that } \sum_{i=1}^{n} \lambda_i = 1.$$

By introducing constants of the form $P(z|D, \theta_n)$ to Equation 3.7 such that $P(z|D, \theta_n) \geq 0$ and $\sum_z P(z|D, \theta_n) = 1$, we can apply Jensen’s inequality as follows:

$$\mathcal{L}(\theta_{n+1}) - \mathcal{L}(\theta_n) = \ln \left( \sum_z P(z|D, \theta_n) \frac{P(D|z, \theta_{n+1})P(z|\theta_{n+1})}{P(z|D, \theta_{n})} \right) - \ln P(D|\theta_n)$$

$$= \ln \left( \sum_z P(z|D, \theta_n) \frac{P(D|z, \theta_{n+1})P(z|\theta_{n+1})}{P(z|D, \theta_{n})} \right) - \ln P(D|\theta_n)$$

$$\geq \sum_z P(z|D, \theta_n) \ln \left( \frac{P(D|z, \theta_{n+1})P(z|\theta_{n+1})}{P(z|D, \theta_{n})} \right) - \ln P(D|\theta_n)$$

$$= \sum_z P(z|D, \theta_n) \ln \left( \frac{P(D|z, \theta_{n+1})P(z|\theta_{n+1})}{P(z|D, \theta_n)P(D|\theta_n)} \right)$$

$$\triangleq \triangle(\theta_{n+1}|\theta_n).$$

(3.8)

Equation 3.8 can be written as

$$\mathcal{L}(\theta_{n+1}) \geq \mathcal{L}(\theta_n) + \triangle(\theta_{n+1}|\theta_n)$$

(3.9)

Equation 3.9 shows that $\mathcal{L}(\theta_n) + \triangle(\theta_{n+1}|\theta_n)$ is bounded above by $\mathcal{L}(\theta_{n+1})$. It is easy to show that any $\theta_{n+1}$ which increases $\mathcal{L}(\theta_n) + \triangle(\theta_{n+1}|\theta_n)$ will also increase $\mathcal{L}(\theta_{n+1})$. 
In order to maximize $L(\theta_{n+1})$, the EM algorithm requires the selection of $\theta_{n+1}$ such that $\mathcal{L}(\theta_n) + \Delta(\theta_{n+1}|\theta_n)$ is maximized. If we denote the maximization value by $\theta'_{n+1}$, then

$$
\theta'_{n+1} = \arg\max_{\theta_{n+1}} \left\{ \mathcal{L}(\theta_n) + \sum_z P(z|D, \theta_n) \ln \frac{P(D|z, \theta_{n+1}) P(z|\theta_{n+1})}{P(D|\theta_{n}) P(z|D, \theta_n)} \right\}
$$

$$
= \arg\max_{\theta_{n+1}} \left\{ \sum_z P(z|D, \theta_n) \ln P(D|z, \theta_{n+1}) P(z|\theta_{n+1}) \right\}
$$

$$
= \arg\max_{\theta_{n+1}} \left\{ \sum_z P(z|D, \theta_n) \ln \frac{P(D, z, \theta_{n+1}) P(z|\theta_{n+1})}{P(D, z|\theta_{n}) P(\theta_{n+1})} \right\}
$$

$$
= \arg\max_{\theta_{n+1}} \left\{ \sum_z P(z|D, \theta_n) \ln P(D, z|\theta_{n+1}) \right\}
$$

$$
= \arg\max_{\theta_{n+1}} \left\{ E_{Z|D, \theta_n} \{ \ln P(D, z|\theta_{n+1}) \} \right\}
$$

Equation 3.10 defines both the expectation and maximization steps. In the expectation step, the algorithm determines the conditional expectation

$$
E_{Z|D, \theta_n} \{ \ln P(D, z|\theta_{n+1}) \};
$$

and in the maximization step, the algorithm maximizes the expression with respect to $\theta_{n+1}$.

The Generalized EM algorithm relaxes the requirement of maximizing $\Delta(\theta_{n+1}|\theta_n)$ to the one of increasing $\Delta(\theta_{n+1}|\theta_n)$ so that $\Delta(\theta'_{n+1}|\theta_n) \geq \Delta(\theta_{n+1}|\theta_n)$. With this requirement it is possible to show that the likelihood $L(\theta_{n+1})$ is guaranteed to be non-decreasing at each iteration.

3.4 Conclusion

The framework of Dynamic Bayesian Networks (DBNs) offers a large space of models which can be exploited to represent and reason about various temporal domains. However, the use of DBNs and Bayesian networks in general is more likely to be successful if the set of variables and the interactions between variables that are defined in the first stages of model construction do provide an adequate representation
of the domain being modeled. Our proposal to use DBNs in transliteration related
tasks can proceed in different ways ranging from determining DBN model structures
given example transliteration data to using already specified DBN structures. In the
thesis we are concerned with the latter where we investigate DBN structures that
have already been proposed but not yet tested in transliteration-related tasks. Given
a DBN model structure, the parameters associated with the model also need to be
estimated before we can use the model to answer queries associated with a given
domain. In this chapter, we have reviewed the expectation maximization algorithm
which has been and still is the cornerstone of parameter estimation in the framework
of DBNs. The estimation of DBN model parameters forms the first main phase in
our application of DBN models in transliteration-related tasks. Given a fully param-
eterized DBN model, we can use a suitable inference algorithm to answer a specific
type of query in the problem domain. In transliteration mining, we would like to use
the model to compute the similarity estimate associated with a source and target
word. Specifically, we want to use the model to compute the probability of observing
the pair as candidate and transliteration. In transliteration generation, we would like
to use the DBN model parameters for suggesting hypothetical target representations
given a source word. The use of a particular inference algorithm to answer a spe-
cific query with regard to transliteration mining and generation constitutes the other
main phase in applying the DBN models.
4.1 Introduction

The detection of transliterations requires an analysis on words written in the source and target languages with the aim of determining word(s) from the target language that are the most likely representation(s) of a word in a source language and vice versa. The transliteration detection process is often performed in two stages: a computation of transliteration similarity between a pair of candidate transliterations, and the decision on whether or not to regard the candidate transliteration pair as a true transliteration pair based on the similarity estimate.

In this chapter, we introduce the approach of Pair Hidden Markov models (Pair HMMs) as the first of two Dynamic Bayesian Network (DBN)-related approaches that we investigate in their application to detecting transliteration pairs from bilingual text. Pair HMMs as the name suggests extend the classic Hidden Markov models (HMMs) by modeling two observation sequences instead of one sequence. Inference using Pair HMMs is also based on modified versions of the classic HMM inference algorithms (Forward-Backward and Viterbi). The Pair HMM approach in its own right offers a huge model space, but we choose to start our investigation with only some model settings that have been successfully used in tasks having requirements similar to those for computing transliteration similarity. Our aim here is to determine whether the assumptions that have been used in constructing Pair HMMs for the related tasks of cognate identification (Mackay and Kondrak 2005) and dialect comparison (Wieling et al. 2007) do hold for the task of detecting transliterations. We then
propose additional Pair HMM parameter settings with the aim of determining the
effect of Pair HMM parameter changes in an experimental transliteration identifica-
tion task. For preliminary experiments we obtained and manually verified geographic
name pairs from the Geonames online database for four language pairs: English-
Dutch, English-French, English-German, and English-Russian. Later, we report on
results from a second set of transliteration identification experiments where we evalu-
ate several Pair HMMs against a standard baseline that uses pair n-gram information
using standard transliteration corpora from the 2009 (Li et al. 2009) and 2010 (Li
et al. 2010) Named entities workshop (NEWS) shared tasks on transliteration genera-
tion. For the second set of experiments, we use seven language pairs: English-Bangla,
English-Chinese, English-Hindi, English-Kannada, English-Russian, English-Tamil,
and English-Thai. In section 4.2 we introduce HMMs and briefly review their recent
application in modeling transliteration. Later, we go on to introduce the approach
of Pair HMMs and describe how we have adapted them to compute transliteration
similarity.

4.2 Hidden Markov Models

4.2.1 A brief review on representation

Hidden Markov models find their origins as extensions to Markov models. Rabiner
(1989) describes how the concept of Markov models can be extended to include the
case where the observation is a probabilistic function of a state, resulting in a model
that is a doubly embedded stochastic process in which the underlying stochastic
process not observable (i.e. it is hidden). Coupled with the defining property that
the underlying stochastic process satisfies the Markov property, we end up with a
Hidden Markov Model. To satisfy the Markov property, the value of the state at
time $t$ (denoted here by $S_t$), is dependent on only the previous state ($S_{t-1}$), and
independent of all other states prior to $t-1$. The outputs from the states also satisfy
the Markov property. That is, the observation in a state at time $t$ (denoted here by
$O_t$) is independent of all other states and observations. Taken together, these Markov
properties lead to the following factorization of the joint distribution of a sequence
of states ($S$) and observations ($O$) (Ghahramani 2001):

$$P(S_1, ..., S_T, O_1, ..., O_T) = P(S_1) P(O_1|S_1) \prod_{t=2}^{T} P(S_t|S_{t-1}) P(O_t|S_t)$$

This factorization of the joint distribution can be represented graphically as a
Dynamic Bayesian network (see Figure 3.2 in Chapter 3). It is clear from equation
4.1 that to determine the probability distribution over sequences of observations, we
need probability distributions over: the initial state $P(S_1)$, the $K \times K$ transition matrix defining $P(S_t|S_{t-1})$, and the output (or emission) model defining $P(O_t|S_t)$.

### 4.2.2 Recent use of HMMs in machine transliteration

The transition and output model $P(O_t|S_t)$ in equation 4.1 can be specified in various ways. In the following, we review some recent formulations for the HMM transition and output models in the context of the transliteration modeling process.

**a) Bi-Stream HMMs**

Zhao et al. (2007) propose a bi-stream HMM for letter-alignment within named entity (NE) pairs. When using the bi-stream HMM, the probability of a source NE (denoted by $s_1^i$) given a target NE (denoted by $t_1^j$), is formulated in equation 4.2 below as:

$$P(s_1^i|t_1^j) = \sum a_i \prod_{i=1}^{l} P(s_i|t_{ai})P(c_{s_i}|c_{t_{ai}})P(a_i|a_{i-1})$$

(4.2)

where $a_i$ maps $s_i$ to the target letter $t_{ai}$ at position $a_i$ in the target NE. $P(a_i|a_{i-1})$ is the transition probability distribution; $P(s_i|t_{ai})$ is a letter-to-letter translation lexicon; $c_{s_i}$ is a letter cluster of $s_i$ and $P(c_{s_i}|c_{t_{ai}})$ is a cluster level translation lexicon. The bi-stream HMM generates two streams of observations: the letters together with their classes following the distribution for $P(s_i|t_{ai})$ and $P(c_{s_i}|c_{t_{ai}})$ at each state respectively. Zhao et al. (2007) also define a constraint to ensure that the transition can only jump forward or stay at the same state.

**b) Maximum n-gram HMMs**

Zhou (2009) views an HMM as a bi-gram model where the transliteration of the current character is dependent on the transliteration of a previous character. In this approach, the underlying hidden process is a sequence of characters in one language which generates characters in the other language. When using the bi-gram HMM, the probability of target NE given the source NE is formulated in equation 4.3 below as:

$$P(t^n_1|s^n_1) = P(t_1|s_1)P(t_2|s_2,t_1)...P(t_n|s_n,t_{n-1})$$

(4.3)

where $P(t_i|s_i) = \frac{\# \text{ of times } s_i \text{ translates to } t_i}{\# \text{ of times } s_i \text{ occurs and}}$ and

$$P(t_i|s_i,t_i-1) = \frac{\# \text{ of times } s_i \text{ translates to } t_i \text{ given } s_{i-1} \rightarrow t_{i-1}}{\# \text{ of times } s_{i-1} \text{ translates to } t_{i-1}}$$
Zhou uses an alignment procedure to build a translation lexicon that is in turn used for obtaining character translation pair occurrences. This facilitates the computation of the probabilities in equation 4.3. It is straightforward to formulate the equations for estimating transliteration probability for the trigram HMM case and other higher order n-gram HMMs.

c) HMMs for searching transliterations

Darwish (2010) uses an approach similar to the bigram HMM approach above. In Darwish’s case, a character sequence (denoted here by \( s'_i \) in the source NE (\( s'_k \)) is taken to be a potential transliteration for a character sequence \( t'_j \) in the target NE (\( t'_l \)). Darwish calculates the probability of \( s'_i \) given \( t'_j \) from a trained model as follows:

\[
P(t'_j|s'_i) = \prod_{s_x...s_y} P(s_x...s_y|t'_k...t'_l) \tag{4.4}
\]

where \( s_x...s_y \) are non overlapping segments generated by finding all \( 2^n - 1 \) segmentations of the character sequence \( s'_i \). According to Darwish (2010), the segmentation producing the highest probability is chosen, and all target segment sequences \( t'_k, ..., t'_l \) that are known to produce \( s_x...s_y \) for each possible segmentation are also produced. If a set of non overlapping sequences of \( t'_k, ..., t'_l \) generates \( t'_j \), then \( t'_j \) is identified as a transliteration of \( s'_i \); and if multiple target sequences have \( P(s'_i|t'_j) > 0 \), then the \( t'_j \) that maximizes \( P(s'_i|t'_j) \) is chosen as the proper transliteration.

There are several other types of HMMs that have been proposed and used successfully on a variety of tasks. In the following section, we introduce the approach of Pair HMMs that is quite different from the HMM approaches reviewed above. As we have already mentioned, our motivation in investigating the Pair HMM approach is based on our observation of its success in various tasks ranging from biological sequence analysis through to the Natural Language Processing (NLP) task of computing word similarity. In section 4.3, we review the origins of the Pair HMM approach, and later, we describe a number of requirements that need to be met for using the approach to compute transliteration similarity.

4.3 Pair Hidden Markov Models

4.3.1 Origins

The approach of Pair HMMs originates from modifications to a pairwise alignment finite state automaton (Durbin et al. 1998). The conversion of the automaton to a Pair HMM is achieved by defining a model which fulfills the representational requirements of HMMs and whose parameters are defined in such a way so as to approximate
the parametric definitions of the finite state automaton. Durbin et al. (1998) define three emission states that correspond to the states of the automaton as follows: a match state (denoted by M) which has emission probability distribution \( p_{ab} \) for emitting an aligned pair of symbols \( a:b \); and two gap states (denoted by X and Y) with distributions \( q_a \) for emitting a symbol \( a \) against a gap. Durbin et al. (1998) also satisfy the requirement that the probabilities for all the transitions leaving each state sum to one. Figure 4.1 illustrates Durbin et al.’s (1998) initial probabilistic version of the pairwise alignment finite state automaton. This probabilistic model is similar to a Hidden Markov model but instead of emitting a single sequence, it emits a pair of sequences.

The model in Figure 4.1 allows for symmetry between source and target and therefore uses two parameters to represent transition probabilities between emission states. These parameters are denoted by \( \delta \) (which represents the transition probability from

![Diagram of probabilistic version of the pairwise alignment finite state automaton](image)

**Figure 4.1:** Probabilistic version of the pairwise alignment finite state automaton in Durbin et al. (1998). Three transition parameters are specified including \( \delta \) (for transitions from the match state (M) to gap states (X or Y)), \( \epsilon \) (for transitions of staying in a gap state, and \( \tau \) (for transitions to the end state).
4. Pair HMMs for machine transliteration

the *match* state to a *gap* state) and $\epsilon$ (which represents the transition probability of staying in a *gap* state). Durbin et al. also defined the start and end states to formalize conditions for initialization and termination. They specifically define transition probabilities from the *start* state to be the same as transition probabilities from the *substitution* state to any of the *emitting* states. Durbin et al. also define the probability of transition to the end state from each *emitting* state ($\tau$) be the same. Durbin et al. (1998) then define the algorithms for inference and learning for the proposed Pair HMM structure in the context of biological sequence analysis. The reader is referred to Durbin et al. (1998) for a detailed description of the accompanying Pair HMM algorithms based on the model structure in Figure 4.1.

### 4.3.2 Pair HMMs for modeling word similarity

After the introduction of the Pair HMM approach, Mackay and Kondrak (2005) proposed to adapt it to compute word similarity with application the cognate identification task. Mackay and Kondrak proposed a number of modifications to Durbin et al.’s (1998) original Pair HMM structure so as to suit the alignment and comparison of words in natural language. In the first modification, they added a pair of transitions between the gap states $X$ and $Y$ each having the same transition probability (denoted by $\lambda$). In another modification, they defined two parameters to represent two different transition probabilities to the end state. They defined $\tau_M$ to represent the transition probability from the *match* state to the end state, and $\tau_{XY}$ to represent the transition probability from the gap states to the end states. Figure 4.2 shows a finite state representation of the Pair HMM proposed by Mackay and Kondrak (2005). Mackay and Kondrak then modified the Pair HMM inference algorithms with regard to the model structure in Figure 4.2. The reader is referred to Mackay’s thesis (Mackay 2004) for a further description of the accompanying algorithms.

### 4.3.3 Pair HMMs for modeling transliteration similarity

The transliteration similarity estimation task is similar to the word similarity estimation task where the Pair HMM method has found successful application (Mackay and Kondrak 2005, Wieling et al. 2007). However, we need to know whether or not the Pair HMM method could be useful for transliteration similarity estimation and ranking. Our starting point is to cast the transliteration similarity estimation task in exactly the same way as the word similarity estimation task. Before we do that, we need to address issues involved in a transliteration similarity estimation task. Some of these issues have been raised by Mackay and Kondrak in the word similarity estimation task:
The first challenge involves dealing with a representation of the source and target language words in a form that enables inference to be done efficiently using Pair HMMs. In transliteration, the source and target language words are transcribed using different writing systems. If we consider the orthographic representation of the words, then our similarity estimation method should be able to deal with the different alphabets. Alternatively, we might use a phonetic alphabet to transcribe the words phonetically. If represented phonetically, then the method will only have to use one alphabet for both source and target languages. Phonetic representation is, however, complicated by the lack of phonetically transcribed data. Moreover, some approaches that consider orthographic-only representations have resulted in comparable if not better machine transliteration performance than phonetic representation approaches.
4. Pair HMMs for machine transliteration

(Li et al. 2004).

The other challenge involves the tokenization of the source and target language words in their representational form into segments that can be used for alignment. A starting point for tokenization, is to use the language’s alphabet (orthographic representation) or a phonetic alphabet (phonetic representation). It is also common to find a combination of characters in one language corresponding to a single character in the other language: an example is the English letter pair ⟨ch⟩ as compared to the Russian representation ⟨ч⟩. In that case it should be possible to represent ⟨ch⟩ as a single token on the English side.

There are some assumptions we can consider to simplify the application of the Pair HMM method to transliteration similarity estimation. We take a leaf from the word similarity estimation task in Mackay (2004) which uses a number these assumptions for applying Pair HMMs in cognate identification. Although the transliteration task involves the analysis of NEs, an NE is viewed here as a ‘type of word’ with respect to a given language. We therefore hardly expect these assumptions to affect the performance of the resulting transliteration system. First, we assume a monotonic ordering of the writing system characters in the source and target language words during transliteration similarity estimation. That is the basic ordering of the tokens remains the same between the source and target language. Second, we assume that there are no crossing links in the alignments. Third, we assume a one-to-one correspondence between alignments. Note that, the third assumption makes the Pair HMM method unsuitable for application to some language pairs whose writing systems are totally different for example between English and Chinese. English uses a phonemic alphabet while the Chinese writing system is mostly logographic. Take for example the name “Peter” as written in English and one of its simplified Chinese representations 彼得. If we tokenize the names by character, and try to get an alignment for them, we will always end up with atmost two character matching alignments while the rest of the characters are aligned to an empty string. Consequently, the resulting Pair HMM will have a poor representation with regard to strings in the source and target language. For this specific case, we would like to use a method that matches the Chinese character ‘彼’ to the English character sequence ‘Pe’, and also matches ‘得’ to ‘ter’ which is close to a true representation of the character correspondences in the source and target language strings.

Mackay and Kondrak (2005) also maintain some symmetries with respect to the gap states “X” and “Y”. The same symmetries are also maintained in (Wieling et al. 2007) in the Dutch dialect comparison task. For the dialect comparison task, it is indeed more meaningful to consider the states “X” and “Y” the same since the source and target words are from the same language. For the transliteration task, where the

\[\text{In a logographic writing system, each symbol in theory represents an idea.}\]
source and target languages use different writing systems, the gap states should be
distinct and should reflect the different properties of the source and target writing
systems. In our preliminary experiments, we determine the effect on the quality
of the transliteration similarity estimate obtained using Pair HMMs that distinguish
between the the gap states with respect to the different writing systems in comparison
to using Pair HMMs where the gap states are assumed to be the same. In the former,
we constrain the the Pair HMMs to generate the pair of words based on the different
probability distributions in the gap states X and Y that reflect the distinct properties
between the source and target writing system respectively. In the latter, the Pair
HMMs generate the pair of words based on a probability distribution for X which is
the same for Y where we assume a single ‘universal’ alphabet which combines both
the source and target writing systems.

When using Pair HMMs, the main focus is usually on the comparison of the
effectiveness of different Pair HMM algorithms and determining the optimal structure
of the underlying model. To determine the optimal structure, we can examine the

\[ \alpha, \beta, \delta \]

Figure 4.3: Finite state representation of a Pair HMM with three transition parameters \((\alpha, \beta, \delta)\). The probability of leaving an edit state to another state is the same for all destination

states.
relative contribution of three sets of parameters (Mackay and Kondrak 2005): substitution parameters, gap parameters (insertion and deletion), and transition parameters. Because substitution parameters constitute the core of a Pair HMM, focus is usually put on the gap and transition parameters. In the second set of Pair HMM experiments, we determine the effect of Pair HMM transition parameters on transliteration identification quality. For our investigation, we have defined three Pair HMM settings in addition to Mackay and Kondrak’s (2005) Pair HMM where we train the models to use different probability distributions for the gap states (X and Y) with regard to the different writing systems for the source and target languages. In the first setting, the Pair HMM is trained without transition parameters between each of the edit states; it only uses transition parameters from a start state to one of the edit states and from one of the edit states to the end state. In the second setting (Figure 4.3), we define the Pair HMM to use three transition parameters associated

![Diagram of Pair HMM with transition probabilities](image)

Figure 4.4: Finite state representation of a Pair HMM with nine distinct transition parameters: six for transition probabilities between the edit states ($\delta_X, \delta_Y, \lambda_X, \lambda_Y, \epsilon_X, \epsilon_Y$), and three to the end state from each edit state ($\tau_X, \tau_Y$, and $\tau_M$).
with leaving one of the edit states, and where we also specify the probability of starting in one of the edit states to be the same as the probability of moving from the substitution state (M) to that edit state including M. In the third setting (Figure 4.4), we define the Pair HMM to use different probabilities for transitions between the edit states and to the end state.

4.3.4 Pair HMMs – Inference

As is the case for the classic HMMs, we are concerned with three important tasks when applying Pair HMMs to transliteration similarity estimation:

The first task, which is also related to our aim for using Pair HMMs, is to compute the probability of a pair of words given a specific Pair HMM. This task requires the use of a Pair HMM algorithm with the corresponding Pair HMM parameters to estimate the similarity and perform a ranking with regard to deciding whether or not the pair of words are likely transliterations. We adapt a number of Pair HMM scoring algorithms that have been applied in related work (Mackay and Kondrak 2005, Wieling et al. 2007) including: Forward, Viterbi, and their corresponding log-odds versions (see below).

The second task is to determine the most probable alignment given a pair of words and a Pair HMM. In this task we need to use a version of the Viterbi algorithm for the given Pair HMM to find the most probable alignment sequence given two words.

The third task is associated with Pair HMM parameter estimation. Given a Pair HMM structure with unspecified parameters, we need to compute Pair HMM parameters that maximize the likelihood of data comprised of pairs of transliteration matches. There are a number of algorithms that we can employ in the training procedure including: The Forward-Backward algorithm and the Viterbi algorithm. In this case, we will use a version of the well known Baum-Welch (Baum et al. 1970) algorithm (also implements a Forward-Backward procedure) for training a given Pair HMM specification.

It is apparent from the tasks above that the core Pair HMM algorithms include: the Forward, Backward and Viterbi. Whenever we decide on a specific Pair HMM, we need to modify the algorithms to suit the structure and semantics of that particular Pair HMM. The reader is referred to (Durbin et al. 1998) and (Mackay 2004) for an introduction and derivation of these Pair HMM algorithms. In the following lines, pseudocode is used to illustrate the Forward, Backward, and Viterbi algorithms corresponding to the distinct Pair HMM. In all cases, we denote the source sequence and target sequence by \(x\) and \(y\) respectively and use \(i\) and \(j\) to denote the indexes in the two sequences respectively.
a) Forward algorithm for the Pair HMM with distinct parameters

The Forward algorithm computes for all possible paths, the total probability of a pair of subsequences \((x_1, ..., x_i, y_1, ..., y_j)\) that have been emitted up to a hidden state \(k\) \((M, X, Y)\). We use the variable \(f^k(i, j)\) to denote this total probability, and \(f^k(i, j)\) to indicate the Forward probability associated with being in any of the states. The initialization, induction, and termination equations for the Forward algorithm specific to the Pair HMM of Figure 4.4 are then written as follows:

\[
\begin{align*}
\text{The forward algorithm for a Pair HMM that uses distinct transition parameters} \\
1. \text{Initialization} & \quad f^M(0,0) = 1 - \delta_X - \delta_Y - \tau_M, \ f^X(0,0) = \delta_X, \ f^Y(0,0) = \delta_Y \\
& \quad \forall f(i,-1) = f^*(-1,j) = 0 \\
2. \text{Induction} & \quad \text{for } 0 \leq i \leq n, \ 0 \leq j \leq m, \ \text{except } (0,0) \text{ do} \\
& \quad f^M(i,j) = p_{x,y}(1 - \delta_X - \delta_Y - \tau_M)f^M(i-1,j-1) + (1 - \epsilon_X - \lambda_X - \tau_X)f^X(i-1,j-1) + (1 - \epsilon_Y - \lambda_Y - \tau_Y)f^Y(i-1,j-1), \\
& \quad f^X(i,j) = q_x[\delta_X f^M(i-1,j) + \epsilon_X f^X(i-1,j) + \lambda_Y f^X(i-1,j)], \\
& \quad f^Y(i,j) = q_y[\delta_Y f^M(i,j-1) + \epsilon_Y f^X(i,j-1) + \lambda_Y f^Y(i,j-1)] \\
& \quad \text{end for} \\
3. \text{Termination} & \quad P(O|\mu) = \tau_M f^M(n,m) + \tau_X f^X(n,m) + \tau_Y f^Y(n,m)
\end{align*}
\]

b) Backward algorithm for the Pair HMM with distinct parameters

In a manner similar to that of the Forward algorithm, the Backward algorithm computes the total probability for all possible paths of the subsequences starting from \(x_{i+1}\) and \(y_{j+1}\) up to the end, when the Pair HMM is in a state \(k\). We use the variable \(b^k(i, j)\) to denote the total Backward probability and \(b(i, j)\) to indicate the Backward probability associated with any of the states. We also use \(n\) and \(m\) to index the end of the source and target sequences respectively. The initialization, induction, and termination equations of the Backward algorithm specific to the Pair HMM of Figure 4.4 are as shown below:

\[
\begin{align*}
\text{The backward algorithm for a Pair HMM that uses distinct transition parameters} \\
1. \text{Initialization} & \quad b^M(n,m) = \tau_M, b^X(n,m) = \tau_X, b^Y(n,m) = \tau_Y \\
2. \text{Induction} & \quad b^M(i,j) = (1 - \delta_X - \delta_Y - \tau_M)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) + \delta_X q_{x_{i+1}}b^X(i+1,j)
\end{align*}
\]
4.3 Pair Hidden Markov Models

The backward algorithm for a Pair HMM that uses distinct transition parameters

\[
\begin{align*}
 b^X(i, j) &= (1 - \epsilon^X - \lambda^X - \tau_M)p_{x_{i+1}y_{j+1}}b^M(i + 1, j + 1) + \epsilon^X y_{j+1} b^X(i + 1, j) \\
&+ \lambda^X y_{j+1} b^Y(i, j + 1), \\
 b^Y(i, j) &= (1 - \epsilon^Y - \lambda^Y - \tau_Y)p_{x_{i+1}y_{j+1}}b^M(i + 1, j + 1) + \lambda^Y y_{j+1} b^Y(i + 1, j) \\
&+ \epsilon^Y y_{j+1} b^Y(i, j + 1).
\end{align*}
\]

3. Termination

\[
P(O|\mu) = (1 - \delta^X - \delta^Y - \tau_M)b^M(0, 0) + \delta^X b^X(0, 0) + \delta^Y b^Y(0, 0)
\]

c) Viterbi algorithm for the Pair HMM with distinct parameters

The Viterbi algorithm is used to find the best alignment sequence(s) given a pair of observation sequences. We use the variable \( v(i, j) \) to denote the probability of emitting the aligned subsequences \( x_1, \ldots, x_i \) and \( y_1, \ldots, y_j \) by the Pair HMM with the sub alignment ending with (a) aligned pair \( x_i \) and \( y_j \) \( (v(i, j) = v^M(i, j)) \), (b) \( x_i \) and \( y_j \) \( (v(i, j) = v^X(i, j)) \), (c) \( x_i \) and \( y_j \) \( (v(i, j) = v^Y(i, j)) \).

The Viterbi algorithm for a Pair HMM that uses distinct transition parameters

1. Initialization

\[
v^M(0, 0) = 1 - \delta^X - \delta^Y - \tau_M, \quad v^X(0, 0) = \delta^X, \quad v^Y(0, 0) = \delta^Y.
\]

All \( v(i, -1) = v(-1, j) = 0 \)

2. Induction

for \( 0 \leq i \leq n \), \( 0 \leq j \leq m \) except \((0, 0)\)

\[
v^M(i, j) = p_{x_{i}y_{j}} \max \left\{ \frac{(1 - \delta^X - \delta^Y - \tau_M)v^M(i - 1, j - 1)}{1 - \epsilon^X - \lambda^X - \tau_M}, \frac{(1 - \epsilon^X - \lambda^X - \tau_M)v^X(i - 1, j - 1)}{1 - \epsilon^X - \lambda^X - \tau_M}\right\},
\]

\[
v^X(i, j) = q_{x_{i}} \max \left\{ \delta^X v^M(i - 1, j), \epsilon^X v^X(i - 1, j), \lambda^X v^X(i - 1, j)\right\},
\]

\[
v^Y(i, j) = q_{y_{j}} \max \left\{ \delta^Y v^M(i, j - 1), \lambda^Y v^Y(i, j - 1), \epsilon^Y v^Y(i, j - 1)\right\},
\]

end for

3. Termination

\[
P(H) = \max(\tau_M v^M(n, m), \tau_X v^X(n, m), \tau_Y v^Y(n, m)).
\]
aligned to an empty string $e$ ($v'(i, j) = v^X(i, j)$), and (c) $y_j$ aligned to the empty string $e$ ($v'(i, j) = v^Y(i, j)$). Using $v'(i, j)$ to indicate the probability in any of the states, the initialization, induction, and termination equations for the specific Pair HMM Viterbi algorithm is as shown above.

d) Log-odds algorithms

The Forward, Backward, and Viterbi Pair HMM algorithms may be sufficient for computing transliteration similarity given a Pair HMM. Durbin et al. (1998) present another Pair HMM algorithm referred to as the log-odds algorithm that takes into account the likelihood of the random occurrence of a pair of observations when computing string similarity. Mackay and Kondrak (2005) show that a Viterbi version of the log-odds algorithm performs significantly better than the other Pair HMM algorithms on the cognate identification task. Since the requirements for computing transliteration similarity are similar to those for computing word similarity where the log-odds algorithm is reported to have been more successful, we also adapt the log-odds algorithm and investigate whether its also valuable for computing transliteration similarity.

The log-odds algorithm uses a random Pair HMM (Figure 4.5) to represent the likelihood of the random occurrence of a pair of strings in the source and target languages. The random Pair HMM does not have a match state since the source and target sequences are assumed to have no underlying relationship to each other. The random model in Figure 4.5 uses only one transition parameter $\eta$ with deletion probabilities ($r_{x_i}$) and insertion probabilities ($r_{y_j}$). Below we briefly review the log-odds algorithm with respect to Mackay and Kondrak’s Pair HMM (Figure 4.2).

The probability of observing a pair of strings using a random Pair HMM $R$ is computed as follows:

$$P(x, y | R) = \eta(1 - \eta)^n \prod_{i=1}^{n} r_{x_i} \eta(1 - \eta)^m \prod_{i=1}^{m} r_{y_j} = \eta^2 (1 - \eta)^{n+m} \prod_{i=1}^{n} r_{x_i} \prod_{i=1}^{m} r_{y_j}. \quad (4.5)$$

where $x$ (having $n$ characters) and $y$ (having $m$ characters) represent source and target strings respectively. An additive model with resulting log-odds emission and transition scores can be specified by combining emission scores and transition scores from the standard and random Pair HMMs. As an example, the following equations are used to merge the emission and transition scores for the Pair HMM with five

---

2There should be no restriction on the number of transition parameters that can be used in the random Pair HMM. It should be possible to use two transition parameters, one associated with deletions, while the other with insertions. However it is the emission states X and Y that contribute more to the final random model probability.
4.3 Pair Hidden Markov Models

Figure 4.5: Finite state representation of the random Pair HMM. This model uses only one transition probability ($\eta$) with deletion probabilities ($r_x$) and insertion probabilities ($r_y$). X and Y nodes respectively refer to deletion and insertion states, and S denotes a start state. The unlabeled node represents a silent state which does not emit any symbols but is used to gather inputs from S and X states. Adapted from Durbin et al (1998).

transition parameters transition parameters (Figure 4.2) to get the standard terms necessary for sequence alignment following a dynamic programming methodology. Let $s(.,.)$ denote the substitution score; $d(.)$ the gap open score; and $e(.)$ and $f(.)$ gap extension scores that correspond to transitions and emissions from the match state to the gap states, and between the gap states respectively:

$$s(x, y) = \log \frac{p_{xy}}{r_x r_y} + \log \frac{1 - 2 \delta - \tau_M}{1 - \eta}$$

$$d(x) = -\log \frac{q_x \delta (1 - \epsilon - \lambda - \tau_{XY})}{r_x (1 - \eta)(1 - 2 \delta - \tau_M)}$$

$$e(x) = -\log \frac{q_x \epsilon}{r_x (1 - \eta)}$$

$$f(x) = -\log \frac{q_x \lambda}{r_x (1 - \eta)}$$

$$c = \log \frac{1 - 2 \delta - \tau_M}{1 - \epsilon - \lambda - \tau_{XY}} + \log(\tau_{XY}).$$

The Pair HMM Viterbi log-odds algorithm for the Pair HMM with five transition parameters is shown on the next page.

The log-odds emission and transition score expressions, and the Viterbi log-odds algorithm for the other Pair HMMs are also derived separately from the corresponding Pair HMMs combined with the Random model. In a similar manner, we derive and apply the log-odds version for the forward algorithm for all the Pair HMM variants.
4. Pair HMMs for machine transliteration

The Viterbi log-odds algorithm for a Pair HMM that uses five transition parameters

1. Initialization
   \( v^M(0, 0) = -2 \log(\eta), \quad v^X(0, 0) = v^Y(0, 0) = -\infty. \)
   All \( v^H(i, -1) = v^X(-1, j) = 0 \)

2. Induction
   \textbf{for} 0 \leq i \leq n, 0 \leq j \leq m \textbf{except} (0, 0) \textbf{do}

   \[ v^M(i, j) = s_{x, y_j} \max \left\{ \begin{array}{l} v^M(i - 1, j - 1) \\ v^X(i - 1, j - 1) \\ v^Y(i - 1, j - 1) \end{array} \right\}, \]

   \[ v^X(i, j) = \max \left\{ \begin{array}{l} v^M(i - 1, j) - d(x_i) \\ v^X(i - 1, j) - e(x_i) \\ v^Y(i - 1, j) - f(x_i) \end{array} \right\}, \]

   \[ v^Y(i, j) = \max \left\{ \begin{array}{l} v^M(i, j - 1) - d(y_j) \\ v^X(i, j - 1) - e(y_j) \\ v^Y(i, j - 1) - f(y_j) \end{array} \right\}. \]

\textbf{end for}

3. Termination
   \[ P(H) = \max(v^M(n, m) + \log(\tau_M), v^X(n, m) + c, v^Y(n, m) + c). \]

4.3.5 Pair HMMs – parameter estimation

A Pair HMM relies on two main sets of parameters for computing string similarity: transition and emission parameters. These parameters need to be estimated before a Pair HMM can be used for calculating a string similarity estimate. There exist different approaches for estimating these parameters; some approaches which are compared in literature include (Arribas-Gil et al. 2006): numerical maximization techniques; Expectation Maximization (EM) algorithm with variants such as stochastic EM and stochastic approximation EM. We do not review the variety of Pair HMM parameter estimation approaches; for the transliteration identification task, we will adopt the Pair HMM EM-based Baum-Welch algorithm which has already found successful application in the cognate identification (Mackay and Kondrak 2005) and dialect comparison (Wieling et al. 2007) tasks.

We begin the description for the Pair HMM Baum-Welch algorithm by first presenting the case for the classic HMMs. The main difference between the Pair HMMs and classic HMMs with respect to parameter estimation is that for the Pair HMMs, we have to consider an extra dimension of observation sequences. Using common
4.3 Pair Hidden Markov Models

notation for the classic HMMs, we specify the transition probability from a state \( s_k \) to a state \( s_l \) at time \( t \) by the variable \( \xi_t(k,l) \) which is formulated as:

\[
\xi_t(k,l) = P(s_k, s_l | O^t, \mu) = \frac{P(s_k, s_l, O^t | \mu)}{P(O^t | \mu)}
\]

where \( O^t \) is an observation sequence and \( \mu \) is a given HMM. Through expansion and simplification using forward \( (f_t(k)) \) and backward \( (b_t(k)) \) variables, it can be shown that

\[
\xi_t(k,l) = f_t(k) P_{kl} e_t(l) \frac{b_{t+1}(l)}{P(O^t | \mu)}
\]

We also specify the probability of being in a state \( k \) at time \( t \) given the observation sequence by the variable \( \gamma_t(k) \):

\[
\gamma_t(k) = P(s_k | O^t, \mu) = \frac{P(s_k, O^t | \mu)}{P(O^t | \mu)} = \frac{f_t(k) b_t(k)}{\sum_{l=1}^N f_t(l) b_t(l)}
\]

It can be seen that

\[
\gamma_t(k) = \sum_{l=1}^N \xi_t(k,l) = \sum_{l=1}^N f_t(k) P_{kl} e_t(l) b_{t+1}(l) \frac{1}{P(O^t | \mu)} = \frac{1}{P(O^t | \mu)} f_t(k) b_t(k)
\]

If we sum over the time index for the two variables \( \xi_t(k,l) \) and \( \gamma_t(k) \), we get expectations (or counts) that can be used in re-estimating the parameters of an HMM using the following equations:

\[
\pi_k = \text{expected number of times in state } k \text{ at time } t = 1 = \gamma_1(k)
\]

\[
P_{kl} = \frac{\text{expected number of transitions from state } k \text{ to state } l}{\text{expected number of transitions from state } k}
\]

\[
= \frac{\sum_{t=1}^{T-1} \xi_t(k,l)}{\sum_{t=1}^{T-1} \gamma_t(k)} = \sum_d \frac{1}{P(O^d)} \sum_t f_t(k) P_{kl} e_t(O^d_{t+1}) b_{t+1}(l)
\]

\[
e_k(v) = \frac{\text{expected number of times in state } k \text{ observing symbol } v}{\text{expected number of times in state } k}
\]
Now, for the Pair HMMs, we have to sum over the positions in each of the sequences, and over all possible sequences. Let $h$ denote the index of the pair of sequences we are using, and let $f$ and $b$ denote the forward and backward variables respectively but this time with an extra dimension. We obtain the following equations for the transition and emission probabilities:

For a transition (k to l) ending in a substitution state we have

$$P_{kl} = \sum_h \frac{1}{P(O|\mu)} \sum_i \sum_j f^h_{(i, j)}(k) P_{kl} e_{(x_{i+1}^h, y_{j+1}^h)}(l)$$

(4.6)

For an emission in the substitution state we have

$$e_k(O_{xy}) = \sum_h \frac{1}{P(O|\mu)} \sum_{i | x_i^h \in O_{xy}} \sum_{j | y_j^h \in O_{xy}} f^h_k(i, j) b^h_k(i, j)$$

(4.7)

The equations for the insertion and deletion states will have slightly different forms. In the insertion state, we only need to match symbol $y_j$. Therefore in the equation for estimating transition probability, we only need to change the index for one of the pairs and we use the emission probability for a symbol from one string against a gap.

The probability estimations for the gap states are as follows:

For the deletion state X:

$$P_{kl} = \sum_h \frac{1}{P(O|\mu)} \sum_i \sum_j f^h_{(i, j)}(k) P_{kl} e_{(y_{j+1}^h)}(l)$$

(4.8)

$$e_k(O_{xy}) = \sum_h \frac{1}{P(O|\mu)} \sum_{i | x_i^h \in O_{xy}} \sum_j f^h_k(i, j) b^h_k(i, j)$$

(4.9)

For the insertion state Y:

$$P_{kl} = \sum_h \frac{1}{P(O|\mu)} \sum_i \sum_j f^h_{(i, j)}(k) P_{kl} e_{(x_{i+1}^h)}(l)$$

(4.10)

$$e_k(O_{xy}) = \sum_h \frac{1}{P(O|\mu)} \sum_{j | y_j^h \in O_{xy}} \sum_i f^h_k(i, j) b^h_k(i, j)$$

(4.11)

These expressions enable the construction of a forward-backward algorithm that we can use to learn all the transition and emission parameters for a Pair HMM.
4.4 Transliteration identification experiments using geographic names data

4.4.1 Data

For the preliminary transliteration identification experiments, we obtained geographic name pairs from the Geonames\textsuperscript{3} database. This database provides a collection of geographic names and how they are alternately represented using different languages. We consider only four language pairs: English-Dutch, English-French, English-German, and English-Russian. As this list of language pairs shows, we also include in our investigation the application of DBN-related models to the case where the source and target language use the same writing system. We propose this as an alternative solution for building bilingual named entity lexicons where names are spelled differently across the languages even for the case of the same writing system.

After extracting the raw collection of geographic name pairs from the Geonames database, we manually checked each dataset (for each language pair) to remove any unwanted entities. For the language pairs where the same writing system is used, we filtered out name pairs where the names have the same spelling. This resulted in much smaller sizes of the datasets for these language pairs. We also found frequent use of diacritics and accents in the datasets. We did not normalize any diacritics to the standard Latin alphabet of 26 characters. The presence of the unusual characters for these language pairs partly justifies our assumption to analyse the data in the context of transliteration since the diacritics and accents convey different pronunciations.

Table 4.1 gives a summary of the dataset sizes after pre-processing.

<table>
<thead>
<tr>
<th>Language pair</th>
<th>Total no. of NE pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-Russian</td>
<td>25104</td>
</tr>
<tr>
<td>English-Dutch</td>
<td>514</td>
</tr>
<tr>
<td>English-French</td>
<td>784</td>
</tr>
<tr>
<td>English-Germany</td>
<td>856</td>
</tr>
</tbody>
</table>

Table 4.1: Total size of cross language name pairs that were obtained from the geonames database

4.4.2 Evaluation setup and results

In this section, we evaluate two Pair HMM settings in an experimental transliteration identification task. In the first setting, we specify Mackay and Kondrak’s Pair HMM

\textsuperscript{3}http://www.geonames.org
to use the same alphabet for the source and target observation sequences. We refer to this Pair HMM setting as PHMM1. In the second setting, we apply Mackay and Kondrak’s Pair HMM with distinct insertion and deletion parameters corresponding to two character vocabularies (one vocabulary for generating source language characters while the other is for generating target language characters). We refer to the second Pair HMM setting as PHMM2. Table 4.2 shows the vocabulary sizes that were obtained from data for the four language pairs with respect to the two Pair HMM experimental settings.

<table>
<thead>
<tr>
<th>Language pair</th>
<th>PHMM1</th>
<th></th>
<th>PHMM1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>English-French</td>
<td>66</td>
<td>57</td>
<td>English</td>
<td>52</td>
</tr>
<tr>
<td>English-Russian</td>
<td>127</td>
<td>91</td>
<td>English</td>
<td>61</td>
</tr>
<tr>
<td>English-Dutch</td>
<td>62</td>
<td>56</td>
<td>English</td>
<td>47</td>
</tr>
<tr>
<td>English-Germany</td>
<td>33</td>
<td>32</td>
<td>English</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 4.2: Total sizes of the alphabets for the language pairs with respect to the two Pair HMM experimental settings

The aim of investigating the two Pair HMM settings at this stage is to determine whether the assumption of generating source and target observations based on a single alphabet is sufficient for transliteration similarity estimation. We also evaluate the use of most of the Pair HMM algorithms presented in Section 4.3.4 for transliteration similarity estimation.

In the previous subsection, the small size of the experimental datasets limits our evaluation of the Pair HMMs. In this chapter, we follow two approaches to address this limitation. In the first approach, we propose to use the transliteration similarity likelihood that each model assigns to correct named entity matches in the transliteration corpus. We therefore apply an information theoretic measure known as corpus cross entropy (CCE) which is computed based on the likelihoods. CCE can be used to evaluate the models without the need for a test corpus. Usually, CCE is used to give an initial idea about how well the models approximate the true representation of data in the corpus. We provide such an evaluation for the English-Russian dataset. However, in order to evaluate the models for identifying transliterations, we need to use standard evaluation metrics. For the second approach, we perform K-fold cross validation where we use the usual standard evaluation metrics of accuracy and mean reciprocal rank (MRR).

Starting with the first approach, we introduce the concept of cross entropy in the context of comparing the models we have proposed to compute transliteration similarity. It is important to note that we provide a rather detailed introduction since there is scarcely a discussion on the use of cross entropy for comparing transliteration
4.4 Transliteration identification experiments using geographic names data

models in the literature. Later, we present our application of corpus cross entropy that we use to compare the two Pair HMM settings including the inference algorithms.

a) Corpus cross entropy

Entropy in information-theoretic terms is used to quantify the uncertainty associated with a random variable. In the context of transliteration, the random variables can be perceived to range over characters or words. If we have a set of events whose probabilities of occurrence are \( P(x_1), P(x_2), ..., P(x_N) \), we may want to know the extent of uncertainty for the events. Such a measure, denoted by \( H(x) \) is defined by Shannon (1948) to have the following properties: (1) \( H(x) \) should be continuous in \( P(x_i) \). (2) If the \( P(x_i) \) are equal, \( P(x_i) = \frac{1}{N} \), then \( H \) should be a monotonic increasing function. (3) if a choice is broken down into two successive choices, the original \( H(x) \) should be the weighted sum of the individual values of \( H \). The only \( H(x) \), that satisfies these three assumptions is specified by the following equation (Shannon 1948):

\[
H(x) = -K \sum_{i=1}^{N} P(x_i) \log P(x_i)
\] (4.12)

where \( K \) is a positive constant. \( H(x) \) is referred to as the entropy of the probability distribution over the events. The choice of the logarithmic base corresponds to the preferred unit for measuring information. For a character sequence \( S = s_1, s_2, ..., s_N \), we can specify the entropy using Equation 4.13.

\[
H(s_1, s_2, ..., s_N) = - \sum_{S_1^N \in A} P(S_1^N) \log P(S_1^N)
\] (4.13)

where \( A \) is the vocabulary of characters for a given language.

The concept of cross entropy comes into play for cases where we use a model distribution (which is a close approximation to \( P \)) instead of the true distribution \( P \). Specifically, cross entropy measures the amount of information needed to represent a test event using the model distribution. The interest usually is to know how well the model distribution approximates \( p \). Equation 4.14 below defines the cross entropy between the model distribution \( m(x_i) \) and the true distribution \( P(x_i) \):

\[
H(p, m) = - \sum_{i} P(x_i) \log m(x_i)
\] (4.14)

One resulting property for cross entropy is that it is an upper bound on the true entropy \( H(P) \). That is, given a model \( m \), \( H(P) \leq H(P, m) \). If \( P = m \), cross entropy is said to be at a minimum and \( H(P, m) = H(P) \). The closer the cross entropy \( H(P, m) \) is to the true entropy \( H(P) \), the better \( m \) becomes an approximation of \( p \).
We can therefore use cross entropy to compare approximate models. Between two models \( m_1 \) and \( m_2 \), the more accurate model should be the one with the lower cross entropy. Here, lower cross entropy means that the result is less surprising since we need very little information to represent the test event. Cross entropy is used to derive the *Perplexity* measure which is used in many fields to measure how well a given model fits the data: \( \text{Perplexity} = 2^{\text{Cross entropy}} \).

We formulate the *cross entropy* involving a Pair HMM that models the similarity between a pair of observation sequences as shown in Equation 4.15 below:

\[
H(P,m) = - \sum_{s \in A_1, t \in A_2} P(s_1 : t_1, ..., s_T : t_T) \log m(s_1 : t_1, ..., s_T : t_T) \tag{4.15}
\]

In this case, we draw the pairs of observations according to the true probability distribution \( P \), but sum the log of their probabilities according to \( m \). If we assume that all the tokens (events) in the corpus are distinct, then the true probability distribution is simply a uniform distribution. Along a different line of reasoning, we can simply use the log probability that the model assigns to the tokens in the corpus for evaluation. This log probability is referred to as the *corpus cross entropy*. Given a corpus \( C \) of size \( N \) consisting of tokens \( c_1, c_2, ..., c_N \), the log probability of a model \( m \) on this corpus is defined by the following equation:

\[
H_C(m) = -(\frac{1}{N}) \times \sum_i \log m(c_i) \tag{4.16}
\]

where the summation is done over the tokens in the corpus. It can be proven that as \( N \) tends to \( \infty \), the corpus cross entropy becomes the cross entropy for the true distribution. To prove the equivalence of the corpus cross entropy with the true entropy, it must be assumed that the corpus has a stationary distribution. The proof depends on the fact that the maximum likelihood estimate approaches the true probability distribution as the size of the corpus tends to \( \infty \).

According to (Manning and Schutze, 2001), it is not exactly correct to use use the result for corpus cross entropy in NLP applications because the stationary distribution assumption is clearly wrong for natural languages. Nonetheless, for a given corpus, we can assume that a language is near enough unchanging and this can be considered as an acceptable approximation to truth (Askari, 2006).

When using CCE, we regard each named entity pair \((s^i, t^i)\) as a token \( c_i \). The CCE of a given Pair HMM \( m \) (CCE\(_m\)) on the corpus is formulated as follows:

\[
\text{CCE}_m = -(\frac{1}{N}) \times \sum_{i=1}^{N} \log(m(s^i, t^i)) \tag{4.17}
\]

where \( m(s^i, t^i) \) is the probability that the Pair HMM \( m \) assigns to a source and target language named entity pair \((s^i, t^i)\).
4.4 Transliteration identification experiments using geographic names data

CCE results on English-Russian geographic names corpus

Here, we divided the English-Russian dataset into two where 90% (22594) of the name pairs were used as training data and the rest as test data. The training data is used to estimate the parameters of the Pair HMMs using the Baum-Welch algorithm. The scoring algorithms for both Pair HMMs (PHMM1 and PHMM2) are then applied to compute a log probability score for each name pair entry in the test set using the parameters of the trained Pair HMMs. We then use Equation 4.17 to compute the CCE for the test. Table 4.3 shows the resulting CCE for the Pair HMM algorithms on the whole English-Russian transliteration corpus.

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>CCE</th>
<th>PHMM1</th>
<th>PHMM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>34.085</td>
<td>23.907</td>
<td></td>
</tr>
<tr>
<td>Forward</td>
<td>33.970</td>
<td>21.201</td>
<td></td>
</tr>
<tr>
<td>Viterbi log-odds (identical)</td>
<td>220.468</td>
<td>110.633</td>
<td></td>
</tr>
<tr>
<td>Viterbi log-odds (distinct)</td>
<td>76.199</td>
<td>48.229</td>
<td></td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>107.927</td>
<td>107.927</td>
<td></td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td>63.353</td>
<td>45.248</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: CCE for the Pair HMM algorithms on the English-Russian test set with 2510 name pairs.

It is clear from Table 4.3 that the use of distinct emission parameters based on two distinct alphabets results in a relatively lower cross entropy (hence less uncertainty) compared to the case where one alphabet is used. It is important to note that we can only use CCE to compare corresponding algorithms of the two Pair HMMs. We cannot use CCE to evaluate the base Pair HMM algorithms with the log-odds algorithms since a base Pair HMM algorithm is also involved in the computation of the log-odds score. This would always result in a higher value of CCE for the log-odds algorithm compared to that for the base Pair HMM algorithm.

We can also compare the two models by looking at the rate of change of CCE as the size of the test set increases. The graphs in Figure 4.6 show the variation of CCE with the size of the English-Russian test set for the different Pair HMM scoring algorithms.

The differences in the curves for the corresponding algorithms in Figure 4.6 also suggest that using two alphabets (PHMM2) generally makes Pair HMMs better at modeling transliteration similarity than when they use one alphabet. However, before we can arrive at a final conclusion, we evaluate the models in a standard transliteration identification setting using two metrics: transliteration identification accuracy.
4. Pair HMMs for machine transliteration

Figure 4.6: Variation of CCE with the English-Russian test set size for different Pair HMM scoring algorithms

and mean reciprocal rank (MRR). In the following subsection, we define the two metrics.

b) Transliteration identification accuracy and mean reciprocal rank

Transliteration identification accuracy measures the proportion of target named entities that are correctly identified as transliteration matches of the respective source
4.4 Transliteration identification experiments using geographic names data

Transliteration identification experiments using geographic names data

Accuracy = \frac{\text{number of correctly identified target named entities}}{\text{total number of test named entities identified}}

Transliteration identification accuracy can be computed at a given cutoff rank. TOP-1 accuracy (or sometimes referred to as \textit{precision at rank 1}) indicates the proportion of named entities in the test set for which the correctly identified target named entity was the first to be returned by the transliteration system. Likewise, TOP-5 accuracy indicates the proportion of test named entities for which the correct transliteration was returned within the first five candidate transliterations. If more than one transliteration is available for a source word in the test data, we need to take the variant transliterations into account. A modified version of the transliteration identification metric will be used in Chapter 7 for this purpose. Equation 4.18 below takes into account the case where we have more than one transliteration for a given source word.

\[ \text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} 1 \text{ if } \exists r_{i,j} = c_{i,1}; \text{ 0 otherwise} \]  

(4.18)

where \( r_{i,j} \) is the \( j \)th reference transliteration for the \( i \)th source word in test data and \( c_{i,1} \) is the first candidate transliteration returned by the transliteration system. \( N \) is the size of the test dataset. For the transliteration identification experiments in this chapter, we assume that there is only one transliteration for each source word. The methods will also be evaluated on only TOP-1 accuracy.

We also use MRR, which is mainly used in information retrieval to evaluate the ranked list of documents returned by a search system. In the context of transliteration identification, MRR is the average of reciprocal ranks associated with the named entities in the test set. A reciprocal rank is the reciprocal of the rank at which the correct target transliteration was identified for a given source named entity.

\[ \text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{R_i} \]  

(4.19)

where \( R_i \) is the rank of the correct transliteration in the returned list of candidate transliterations for the \( i \)th test named entity. \( N \) is the total number of named entities in the test set.

When using these metrics for the preliminary set of experiments, we perform \textit{stratified K-fold cross-validation} for evaluation. Equations 4.18 and 4.19 are used for computing the accuracy and MRR respectively in each fold; the cross validation accuracy (CVA) and cross validation MRR (CVMRR) are computed by averaging...
the $K$ individual accuracy and MRR values from each of the $K$ folds.

\[
\text{CVA} = \frac{1}{K} \sum_{k=1}^{K} A_k. \tag{4.20}
\]

where $A_k$ is the accuracy of a model on the $k^{th}$ test data set (see Equation 4.18) while $K$ is the total number of test datasets for cross-validation. Likewise,

\[
\text{CVMRR} = \frac{1}{K} \sum_{k=1}^{K} \text{MRR}_k \tag{4.21}
\]

where MRR$_k$ is the MRR of a model on the $k^{th}$ test data set (see Equation 4.19).

Transliteration identification accuracy and MRR results

Here, we evaluate the models on all the four language pairs for which we perform stratified 10-fold cross validation. During training, a Pair HMM Baum-Welch algorithm is used to estimate the emission and transition parameters of the corresponding Pair HMM. Table 4.4 shows the average number of iterations that were required for the Baum-Welch algorithm to converge for three of the four language pairs.

<table>
<thead>
<tr>
<th>Language pair</th>
<th>avg. no. of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PHMM1</td>
</tr>
<tr>
<td>English-Russian</td>
<td>802</td>
</tr>
<tr>
<td>English-Dutch</td>
<td>896</td>
</tr>
<tr>
<td>English-Germany</td>
<td>418</td>
</tr>
</tbody>
</table>

Table 4.4: Average number of iterations required to converge to local maximum by PHMM1 and PHMM2 Baum-Welch algorithms.

Table 4.4 suggests that Baum-Welch algorithm converges faster to a local optimum for the case of PHMM2 than the case for PHMM1 on all the language pairs. Table 4.5 shows the CVA and CVMRR results from the use of the scoring algorithms for the two Pair HMMs on the English-French dataset.

The results in Table 4.5 suggest that if the standard Pair HMM algorithms are used for computing transliteration similarity, there is no gain in using two separate alphabets between English and French. Although all the log-odds versions of the standard algorithms for PHMM2 achieve a slight improvement in transliteration identification quality compared to those for PHMM1, the differences in CVA and CVMRR are not significant. Table 4.5 also shows that the log-odds algorithms that
4.4 Transliteration identification experiments using geographic names data

Table 4.5: CVA and CVMRR transliteration identification results for different Pair HMM algorithms on English-French data. Values in **bold** indicate the best result.

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>CVA PHMM1</th>
<th>CVA PHMM2</th>
<th>CVMRR PHMM1</th>
<th>CVMRR PHMM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.836</td>
<td>0.824</td>
<td>0.858</td>
<td>0.850</td>
</tr>
<tr>
<td>Forward</td>
<td>0.834</td>
<td>0.827</td>
<td>0.857</td>
<td>0.852</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.675</td>
<td>0.682</td>
<td>0.751</td>
<td>0.752</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.835</td>
<td>0.844</td>
<td>0.861</td>
<td>0.866</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.676</td>
<td>0.677</td>
<td>0.751</td>
<td>0.751</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.845</strong></td>
<td><strong>0.859</strong></td>
<td><strong>0.866</strong></td>
<td><strong>0.875</strong></td>
</tr>
</tbody>
</table>

use distinct emission parameters result in better transliteration identification quality than the standard algorithms for both Pair HMMs.

Table 4.6 shows the CVA and CVMRR results from the application of PHMM1 and PHMM2 on the English-Russian dataset.

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>CVA PHMM1</th>
<th>CVA PHMM2</th>
<th>CVMRR PHMM1</th>
<th>CVMRR PHMM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.759</td>
<td>0.824</td>
<td>0.788</td>
<td>0.834</td>
</tr>
<tr>
<td>Forward</td>
<td>0.759</td>
<td>0.830</td>
<td>0.788</td>
<td>0.838</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.619</td>
<td>0.722</td>
<td>0.680</td>
<td>0.770</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.696</td>
<td>0.824</td>
<td>0.740</td>
<td>0.835</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.749</td>
<td>0.750</td>
<td>0.790</td>
<td>0.790</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.831</strong></td>
<td><strong>0.833</strong></td>
<td><strong>0.840</strong></td>
<td><strong>0.841</strong></td>
</tr>
</tbody>
</table>

Table 4.6: CVA and CVMRR transliteration identification results for two Pair HMMs and different scoring algorithms on English-Russian Geonames data.

In Table 4.6, we see a general improvement in transliteration identification quality from PHMM1 to PHMM2. Specifically, we see a considerable improvement in CVA and CVMRR for the standard Pair HMM algorithms. Again, The Forward log-odds algorithm that uses distinct emission parameters for the random and standard Pair HMMs results in relatively high transliteration identification quality compared to other scoring algorithms although with a slight difference compared to the standard Pair HMM algorithms in PHMM2. There is also a bigger difference between PHMM1 and PHMM2 while using the Forward log-odds as compared to the case for English-French.

Table 4.7 shows the CVA and CVMRR results from the application of PHMM1
and PHMM2 on the English-Dutch dataset.

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>CVA PHMM1</th>
<th>PHMM2</th>
<th>CVMRR PHMM1</th>
<th>PHMM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.804</td>
<td>0.796</td>
<td>0.836</td>
<td>0.829</td>
</tr>
<tr>
<td>Forward</td>
<td>0.806</td>
<td>0.808</td>
<td>0.836</td>
<td>0.838</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.781</td>
<td>0.779</td>
<td>0.821</td>
<td>0.817</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.824</td>
<td>0.826</td>
<td>0.846</td>
<td>0.846</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.783</td>
<td>0.783</td>
<td>0.821</td>
<td>0.821</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td>0.820</td>
<td>0.824</td>
<td>0.845</td>
<td>0.847</td>
</tr>
</tbody>
</table>

Table 4.7: CVA and CVMRR transliteration identification results for two Pair HMMs and different scoring algorithms on English-Dutch Geonames data.

The results in Table 4.7 suggest that the Viterbi log-odds algorithm achieves a relatively higher transliteration identification accuracy compared to the other Pair HMM algorithms although just slightly higher as compared to the forward log-odds algorithm. Table 4.7 results also show that not all algorithms result in improved transliteration identification quality when used for PHMM2. Generally, the differences in CVA and CVMRR between the corresponding PHMM1 and PHMM2 algorithms are so small.

Table 4.8 shows the CVA and CVMRR results from the application of PHMM1 and PHMM2 on the English-German dataset.

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>CVA PHMM1</th>
<th>PHMM2</th>
<th>CVMRR PHMM1</th>
<th>PHMM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.893</td>
<td>0.895</td>
<td>0.920</td>
<td>0.919</td>
</tr>
<tr>
<td>Forward</td>
<td>0.898</td>
<td>0.896</td>
<td>0.922</td>
<td>0.922</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.802</td>
<td>0.805</td>
<td>0.859</td>
<td>0.860</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.829</td>
<td>0.920</td>
<td>0.938</td>
<td>0.938</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.808</td>
<td>0.808</td>
<td>0.865</td>
<td>0.865</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td>0.914</td>
<td>0.911</td>
<td>0.935</td>
<td>0.934</td>
</tr>
</tbody>
</table>

Table 4.8: CVA and CVMRR transliteration identification results for two Pair HMMs and different scoring algorithms on English-German Geonames data.

Just like in the results for the English-Dutch dataset, Table 4.8 shows that there is hardly any difference in using PHMM1 compared to using PHMM2. Again the log-odds algorithms that use distinct emission parameters achieve higher transliteration identification accuracy compared to the base algorithms.
c) Discussion

From the results on the four language pairs (that is English-French, English-Russian, English-Dutch and English-German), we saw some variations in transliteration identification quality on using the different Pair HMM algorithms. The log-odds algorithms (that use distinct emission parameters between the random and standard Pair HMM) consistently performed well on all datasets based on the CVA and CVMRR metrics. The use of an information theoretic metric (*corpus cross entropy*) also suggested that PHMM2 algorithms are better approximators of transliteration similarity than PHMM1 algorithms on the English-Russian dataset. The results also suggest that some algorithms seem to be insensitive to changes in model settings, for example, the Forward log-odds algorithm resulted in almost the same transliteration identification quality under the two Pair HMM settings while the transliteration identification quality for the Forward and Viterbi Pair HMM algorithms changed considerably. The results from the experimental transliteration identification task also differ from those reported in previous work. In the cognate recognition task in (Mackay and Kondrak 2005) and the Dutch dialect comparison task (Wieling et al. 2007), the Viterbi log-odds algorithm is reported to have performed better than all the other algorithms on nine language pairs that were used. This was also the case for the individual language pair of English-French in the cognate identification task. However, for English-French and English-Russian, the Forward log-odds algorithm performed consistently better. This seems to suggest that the properties associated with transliteration data may indeed differ from those of the datasets in tasks similar to transliteration identification thus necessitating a check on various model settings. We specifically see bigger differences in CVA and CVMRR between PHMM1 and PHMM2 for the English-Russian dataset where the languages indeed use different writing systems than the case for the other language pairs where the languages use the same writing system.

4.5 Experiments using NEWS 2009 and 2010 shared task data

4.5.1 Data

For the transliteration identification experiments in this section, we use manually verified transliteration data from the NEWS 2009 (Li et al. 2009) and 2010 (Li et al. 2010) shared tasks on transliteration generation. The transliteration data has been made available as standard data for evaluating machine transliteration systems. We evaluate the transliteration identification methods on seven language
76

4. Pair HMMs for machine transliteration

pairs: English-Bangla, English-Chinese, English-Hindi, English-Kannada, English-
Russian, English-Tamil and English-Thai. For some language pairs, the respective
shared task datasets did not need any further processing before being used. However,
for some language pairs, the datasets needed some pre-processing which we describe
in the following.

NE pairs that had spaces in them and of which the total size of the constituent
names was equal on the source and target side, were split into single NE pairs. We
assumed a monotonic ordering on the space-separated NEs and matched single names
by their corresponding position. It was common to find the use of a comma in the
space-separated NEs, especially in person names (for example ‘WATSON, JOHN B.’
in English and ‘วัตสัน, จอห/uni0E4C.lowน บี.’ in Thai). The comma and the ordering in which
the names were written was always the same on both the source and target side. From
the English-Thai example above, the name ‘วัตสัน’ in the first position in the
Thai NE is indeed a true Thai transliteration of the name ‘WATSON’ which is also
in the first position in the English NE, and the same is true for the two remaining
strings in the respective positions of the complete NEs. We could as well have left out
the space-separated NE pairs but a manual verification\(^4\) on the split-matched NEs
showed that almost all of the mappings were correct. We also removed any NE pairs
that had unnecessary representations in them including: numerical representations,
abbreviations, and person titles (such as Mrs., Mr., Dr., etc.). Table 4.9 shows the
total number of single NE pairs that were used per language pair.

<table>
<thead>
<tr>
<th>Language pair</th>
<th>Total size</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-Bangla</td>
<td>14255</td>
<td>12814</td>
<td>1441</td>
</tr>
<tr>
<td>English-Chinese</td>
<td>37228</td>
<td>33505</td>
<td>3723</td>
</tr>
<tr>
<td>English-Hindi</td>
<td>16059</td>
<td>14486</td>
<td>1573</td>
</tr>
<tr>
<td>English-Kannada</td>
<td>14368</td>
<td>12940</td>
<td>1428</td>
</tr>
<tr>
<td>English-Russian</td>
<td>7840</td>
<td>7056</td>
<td>784</td>
</tr>
<tr>
<td>English-Tamil</td>
<td>14622</td>
<td>13164</td>
<td>1458</td>
</tr>
<tr>
<td>English-Thai</td>
<td>29050</td>
<td>26126</td>
<td>2924</td>
</tr>
</tbody>
</table>

Table 4.9: Number of named entities from NEWS 2010 shared task standard transliteration
data for Training and Testing per language pair.

The English-Chinese datasets were first transformed into a Pinyin representation
ignoring tones. A preliminary run on the original Chinese orthographic representation
using all the Pair HMM variants proposed in this chapter resulted in very low translit-

\(^4\)Verification in this case was possible since this the total size of this class of space separated NE
pairs did not exceed 500 on all language pairs.
eration identification accuracy. However, based on recent literature (Jiampojamarn et al. 2010, Zhao et al. 2007, Zhou 2009), a plausible approach for Chinese involves transforming the characters into Latin character representation using a Romanization system such as Pinyin to simplify transliteration analysis. For example, the Chinese characters in 彼得 would be transformed as follows using the Pinyin Romanization system: ‘彼’→bi’ and ‘得’→de’ (assuming we ignore tones). With such a transformation, we have the option of using the same vocabulary to represent the English and Chinese strings when applying the Pair HMMs. However, we specify the models to use distinct vocabularies with respect to the writing systems. In English-Chinese transliteration identification, it is also sufficient to use the Romanized form for analysis, but in a transliteration generation task where the target string should be a Chinese string, we would need to transform the Romanized representation into Chinese characters. Finally, for some language pairs, the named entities were already sorted alphabetically on the English side and had to be randomized before dividing the whole set into train and test sets.

4.5.2 Evaluation setup and results

For this set of experiments, we report on results for only one held-out test set for each language pair; we therefore evaluate the models using transliteration identification accuracy (Equation 4.18) and mean reciprocal rank (Equation 4.19). In this section, we evaluate the Pair HMMs against a standard baseline of using pair n-gram information.

a) Baseline – Pair n-gram models

We use as a baseline, a method that relies on the correspondence of source and target n-grams for estimating transliteration similarity. In its own right, this baseline method has a large model space for which we can derive various probabilistic and non-probabilistic similarity estimation schemes.

Consider a pair tri-gram case between English and Russian. To describe the method, we use the following name pair “Peter” (English) and “пётр” (Russian) which we assume exists in training data. We would like to elicit tri-gram relationships using the name pair by identifying tri-gram correspondences at the same position in the source and target. To ensure that we do not miss out on any characters in the source

---

5This would require another step in the process of transliteration generation, that is, the segmentation of the string in the source language to enable mapping to a Romanized form for converting to Chinese characters.

6We also performed 10-fold cross validation but there was hardly any error of margin to using only one held-out test set. We report on results for only one held-out test set since we shall use them for comparison purposes with several other models in the next chapter.
or target during the tri-gram correspondence identification process, we introduce a ‘dummy’ character denoted by # which we add at the start and end of each name as follows:

Peter → {##P, #Pe, Pet, etc, ter, er#, r##}

пётр → {##п, #пё, пёт, тр, тр#, р##}

First, we move in the forward direction, while storing information about the tri-gram correspondences. For the name pair example above, we obtain the following correspondences in the forward direction:

Tri-gram pairsforward = {##P-##п, #Pe-#пё, Pet-пёт, etc-тр, ter-тр#, er#-р##}

After the forward pass, we move in the backward direction resulting in the following tri-gram correspondences:

Tri-gram pairsbackward = {r##-р##, er#-тр#, ter-трp#, etc-nёr, Pet-#nё, #Pe-##n}

The transliteration similarity scheme we use in this approach is based on whether or not a tri-gram correspondence associated with a candidate pair in the test set matches any of the tri-gram correspondences identified from the name pairs in the train set. If it does match, then the transliteration similarity score for the candidate pair is incremented by 1 otherwise we evaluate the next trigram pair until the end.

Apart from tri-gram relationships, we also explore bi-gram, 4-gram, 5-gram, and 6-gram relationships in a similar manner.

We have applied this n-gram approach to transliteration data for the seven language pairs. Table 4.10 shows the transliteration identification accuracy and mean reciprocal rank for the different pair n-gram models. As can be seen in Table 4.10, the accuracy and MRR for the pair tri-gram model and the other higher order pair n-gram models are already high for five of the language pairs. As mentioned before, the pair n-gram approach can be associated with a variety of scoring schemes. In a stratified 10-fold cross validation transliteration identification experiment, we found that the use of the scoring scheme described in the last paragraph led to considerably better transliteration identification accuracy and MRR than the case when we use direct probabilities computed for the n-gram correspondences in the train set. Table 4.10 also shows that apart from the case for English-Chinese, the pair tri-gram model is generally better at modeling transliteration similarity in comparison to the other pair n-gram models.

b) Pair HMM results

For this set of experiments we also trained each Pair HMM variant using the Baum-Welch EM algorithm starting with uniform initial distributions for starting, emission,
Table 4.10: Transliteration identification accuracy and MRR for different pair n-gram models on seven language-pairs. Bold values indicate best results for each language pair.

<table>
<thead>
<tr>
<th>Language Pair</th>
<th>bigram</th>
<th>trigram</th>
<th>4-gram</th>
<th>5-gram</th>
<th>6-gram</th>
<th>4-gram</th>
<th>5-gram</th>
<th>6-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>En-Ba</td>
<td>0.198</td>
<td>0.804</td>
<td>0.767</td>
<td>0.724</td>
<td>0.728</td>
<td>0.767</td>
<td>0.771</td>
<td>0.785</td>
</tr>
<tr>
<td>En-Ch</td>
<td>0.017</td>
<td>0.524</td>
<td>0.809</td>
<td>0.632</td>
<td>0.614</td>
<td>0.678</td>
<td>0.683</td>
<td>0.675</td>
</tr>
<tr>
<td>En-Hi</td>
<td>0.393</td>
<td>0.813</td>
<td>0.809</td>
<td>0.802</td>
<td>0.806</td>
<td>0.809</td>
<td>0.837</td>
<td>0.845</td>
</tr>
<tr>
<td>En-Ka</td>
<td>0.283</td>
<td><strong>0.763</strong></td>
<td>0.738</td>
<td>0.708</td>
<td>0.710</td>
<td>0.738</td>
<td>0.751</td>
<td>0.755</td>
</tr>
<tr>
<td>En-Ru</td>
<td>0.837</td>
<td><strong>0.944</strong></td>
<td>0.922</td>
<td>0.903</td>
<td>0.903</td>
<td>0.922</td>
<td>0.920</td>
<td>0.925</td>
</tr>
<tr>
<td>En-Ta</td>
<td>0.138</td>
<td><strong>0.775</strong></td>
<td>0.762</td>
<td>0.725</td>
<td>0.705</td>
<td>0.762</td>
<td>0.768</td>
<td>0.759</td>
</tr>
<tr>
<td>En-Th</td>
<td>0.142</td>
<td>0.590</td>
<td><strong>0.623</strong></td>
<td>0.588</td>
<td>0.549</td>
<td>0.623</td>
<td>0.645</td>
<td>0.609</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>accuracy</th>
<th>MRR</th>
<th>accuracy</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>bigram</td>
<td>0.252</td>
<td>0.262</td>
<td>0.453</td>
<td>0.341</td>
</tr>
<tr>
<td>trigram</td>
<td><strong>0.844</strong></td>
<td><strong>0.604</strong></td>
<td><strong>0.850</strong></td>
<td><strong>0.806</strong></td>
</tr>
<tr>
<td>4-gram</td>
<td>0.811</td>
<td><strong>0.736</strong></td>
<td>0.843</td>
<td>0.778</td>
</tr>
<tr>
<td>5-gram</td>
<td>0.771</td>
<td>0.683</td>
<td>0.837</td>
<td>0.751</td>
</tr>
<tr>
<td>6-gram</td>
<td>0.785</td>
<td>0.675</td>
<td>0.845</td>
<td>0.755</td>
</tr>
</tbody>
</table>

and transition parameters. The trained models were then evaluated on transliteration identification accuracy and MRR. Tables 4.11 to 4.17 show the transliteration identification accuracy and MRR results for different scoring algorithms (with respect to each Pair HMM variant) on test data for the seven language pairs. In the Tables, PHMM0 denotes the Pair HMM variant which is specified and trained with only emission parameters and no transition parameters between edit states; PHMM3 denotes the Pair HMM which is specified and trained to use three transition parameters between edit states (Figure 4.3); PHMM5 denotes the Pair HMM which uses five transition parameters between the edit states and distinct emission parameters in all the edit states (Figure 4.2); and PHMM9 denotes the Pair HMM that uses nine distinct transition parameters between the edit states in addition to distinct emission parameters in each edit state (Figure 4.4). The accuracy and MRR values in bold indicate the best result for a given Pair HMM and when in bold and italicized, then the result is the overall best for that language pair. As almost all tables show, the Forward log-odds algorithm (where the random model’s insertion and deletion parameters are based on character frequencies from the respective training data) for each Pair HMM variant consistently and in most cases outperforms the other algorithms for all the seven language pairs. On the other hand, we see that the Forward log-odds and Viterbi log-odds (that use the same insertion and deletion parameters for the standard models and the random model), consistently result in poor
### 4. Pair HMMs for machine transliteration

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>PHMM0</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.717</td>
<td>0.579</td>
<td>0.776</td>
<td>0.795</td>
<td>0.773</td>
</tr>
<tr>
<td>Forward</td>
<td>0.822</td>
<td>0.731</td>
<td>0.869</td>
<td>0.862</td>
<td>0.883</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.462</td>
<td>0.523</td>
<td>0.524</td>
<td>0.552</td>
<td>0.583</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.877</td>
<td>0.754</td>
<td>0.783</td>
<td>0.832</td>
<td>0.918</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.501</td>
<td>0.614</td>
<td>0.571</td>
<td>0.588</td>
<td>0.621</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.902</strong></td>
<td><strong>0.932</strong></td>
<td><strong>0.873</strong></td>
<td><strong>0.912</strong></td>
<td>0.936</td>
</tr>
<tr>
<td>Pair tri-gram baseline</td>
<td>0.804</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Viterbi                 | 0.773 | 0.655 | 0.832 | 0.844 |
| Forward                 | 0.883 | 0.811 | 0.906 | 0.897 |
| Vit. log-odds (identical) | 0.583 | 0.631 | 0.625 | 0.659 |
| Vit. log-odds (distinct) | 0.918 | 0.803 | 0.835 | 0.878 |
| Forward log-odds (identical) | 0.621 | 0.737 | 0.678 | 0.704 |
| Forward log-odds (distinct) | **0.936** | **0.950** | **0.911** | **0.937** |
| Pair tri-gram baseline  | 0.844 |

Table 4.11: *English-Bangla transliteration identification accuracy and MRR for Pair HMMs*

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>PHMM0</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.205</td>
<td>0.111</td>
<td>0.418</td>
<td>0.426</td>
<td></td>
</tr>
<tr>
<td>Forward</td>
<td>0.293</td>
<td>0.236</td>
<td>0.482</td>
<td>0.487</td>
<td></td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.050</td>
<td>0.080</td>
<td>0.132</td>
<td>0.132</td>
<td></td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.180</td>
<td>0.237</td>
<td>0.537</td>
<td>0.537</td>
<td></td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.086</td>
<td>0.129</td>
<td>0.145</td>
<td>0.144</td>
<td></td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.467</strong></td>
<td><strong>0.684</strong></td>
<td><strong>0.666</strong></td>
<td><strong>0.661</strong></td>
<td></td>
</tr>
<tr>
<td>Pair tri-gram baseline</td>
<td>0.524</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Viterbi                 | 0.286 | 0.163 | 0.533 | 0.538 |
| Forward                 | 0.411 | 0.358 | 0.611 | 0.615 |
| Vit. log-odds (identical) | 0.088 | 0.126 | 0.209 | 0.209 |
| Vit. log-odds (distinct) | 0.233 | 0.296 | 0.634 | 0.632 |
| Forward log-odds (identical) | 0.150 | 0.205 | 0.228 | 0.226 |
| Forward log-odds (distinct) | **0.586** | **0.777** | **0.764** | **0.760** |
| Pair tri-gram baseline  | 0.604 |

Table 4.12: *English-Chinese Pair HMM transliteration identification accuracy and MRR*
### 4.5 Experiments using NEWS 2009 and 2010 shared task data

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>PHMM0</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>accuracy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Viterbi</td>
<td>0.639</td>
<td>0.651</td>
<td>0.671</td>
<td>0.733</td>
</tr>
<tr>
<td>Forward</td>
<td>0.733</td>
<td>0.755</td>
<td><strong>0.804</strong></td>
<td>0.807</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.284</td>
<td>0.464</td>
<td>0.253</td>
<td>0.349</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.585</td>
<td>0.787</td>
<td>0.636</td>
<td>0.729</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.429</td>
<td>0.553</td>
<td>0.307</td>
<td>0.426</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.843</strong></td>
<td><strong>0.885</strong></td>
<td>0.793</td>
<td><strong>0.870</strong></td>
</tr>
<tr>
<td><strong>Pair tri-gram baseline</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>0.813</strong></td>
</tr>
<tr>
<td><strong>MRR</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Viterbi</td>
<td>0.710</td>
<td>0.745</td>
<td>0.754</td>
<td>0.808</td>
</tr>
<tr>
<td>Forward</td>
<td>0.817</td>
<td>0.831</td>
<td><strong>0.866</strong></td>
<td>0.867</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.396</td>
<td>0.576</td>
<td>0.345</td>
<td>0.458</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.683</td>
<td>0.853</td>
<td>0.724</td>
<td>0.805</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.549</td>
<td>0.665</td>
<td>0.414</td>
<td>0.538</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.899</strong></td>
<td><strong>0.923</strong></td>
<td>0.861</td>
<td><strong>0.911</strong></td>
</tr>
</tbody>
</table>

**Table 4.13:** English-Hindi Pair HMM transliteration identification accuracy and MRR

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>PHMM0</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>accuracy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Viterbi</td>
<td>0.593</td>
<td>0.602</td>
<td>0.780</td>
<td>0.758</td>
</tr>
<tr>
<td>Forward</td>
<td>0.636</td>
<td>0.693</td>
<td>0.816</td>
<td>0.841</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.342</td>
<td>0.422</td>
<td>0.426</td>
<td>0.411</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.698</td>
<td>0.769</td>
<td>0.801</td>
<td>0.762</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.326</td>
<td>0.447</td>
<td>0.490</td>
<td>0.479</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.779</strong></td>
<td><strong>0.855</strong></td>
<td><strong>0.858</strong></td>
<td><strong>0.854</strong></td>
</tr>
<tr>
<td><strong>Pair tri-gram baseline</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>0.763</strong></td>
</tr>
<tr>
<td><strong>MRR</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Viterbi</td>
<td>0.659</td>
<td>0.701</td>
<td>0.849</td>
<td>0.831</td>
</tr>
<tr>
<td>Forward</td>
<td>0.746</td>
<td>0.790</td>
<td>0.873</td>
<td>0.891</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.460</td>
<td>0.535</td>
<td>0.546</td>
<td>0.526</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.782</td>
<td>0.836</td>
<td>0.862</td>
<td>0.831</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.448</td>
<td>0.574</td>
<td>0.608</td>
<td>0.603</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.858</strong></td>
<td><strong>0.904</strong></td>
<td><strong>0.905</strong></td>
<td><strong>0.903</strong></td>
</tr>
</tbody>
</table>

**Table 4.14:** English-Kannada Pair HMM transliteration identification accuracy and MRR
### 4. Pair HMMs for machine transliteration

#### Table 4.15: English-Russian Pair HMM transliteration identification accuracy and MRR

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>PHMM0</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.688</td>
<td>0.781</td>
<td>0.876</td>
<td>0.878</td>
</tr>
<tr>
<td>Forward</td>
<td>0.810</td>
<td>0.818</td>
<td>0.881</td>
<td>0.876</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.365</td>
<td>0.740</td>
<td>0.823</td>
<td>0.828</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.633</td>
<td>0.866</td>
<td>0.878</td>
<td>0.884</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.472</td>
<td>0.731</td>
<td>0.836</td>
<td>0.841</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.864</strong></td>
<td><strong>0.885</strong></td>
<td><strong>0.888</strong></td>
<td><strong>0.887</strong></td>
</tr>
<tr>
<td>Pair tri-gram baseline</td>
<td>0.944</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Viterbi                    | 0.738 | 0.821 | 0.884 | 0.885 |
| Forward                    | 0.843 | 0.846 | 0.888 | 0.885 |
| Vit. log-odds (identical)  | 0.442 | 0.787 | 0.850 | 0.854 |
| Vit. log-odds (distinct)   | 0.565 | 0.786 | 0.861 | 0.863 |
| Forward log-odds (identical)| **0.878** | **0.889** | **0.892** | **0.891** |
| Forward log-odds (distinct) |       |       |       |       |
| Pair tri-gram baseline     | 0.956 |       |       |       |

#### Table 4.16: English-Tamil Pair HMM transliteration identification accuracy and MRR

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>PHMM0</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.447</td>
<td>0.558</td>
<td>0.681</td>
<td>0.652</td>
</tr>
<tr>
<td>Forward</td>
<td>0.559</td>
<td>0.679</td>
<td>0.766</td>
<td>0.773</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.227</td>
<td>0.347</td>
<td>0.227</td>
<td>0.190</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.649</td>
<td>0.726</td>
<td>0.709</td>
<td>0.660</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.201</td>
<td>0.383</td>
<td>0.281</td>
<td>0.223</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.702</strong></td>
<td><strong>0.824</strong></td>
<td><strong>0.827</strong></td>
<td><strong>0.789</strong></td>
</tr>
<tr>
<td>Pair tri-gram baseline</td>
<td>0.775</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Viterbi                    | 0.529 | 0.670 | 0.771 | 0.741 |
| Forward                    | 0.684 | 0.780 | 0.843 | 0.848 |
| Vit. log-odds (identical)  | 0.342 | 0.463 | 0.346 | 0.278 |
| Vit. log-odds (distinct)   | 0.745 | 0.810 | 0.796 | 0.746 |
| Forward log-odds (identical)| 0.315 | 0.506 | 0.420 | 0.320 |
| Forward log-odds (distinct) | **0.799** | **0.889** | **0.886** | **0.861** |
| Pair tri-gram baseline     |       |       |       | 0.824 |
4.5 Experiments using NEWS 2009 and 2010 shared task data

<table>
<thead>
<tr>
<th>Pair HMM algorithm</th>
<th>PHMM0</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>accuracy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Viterbi</td>
<td>0.390</td>
<td>0.348</td>
<td>0.468</td>
<td>0.474</td>
</tr>
<tr>
<td>Forward</td>
<td>0.490</td>
<td>0.458</td>
<td>0.518</td>
<td>0.520</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.141</td>
<td>0.165</td>
<td>0.150</td>
<td>0.156</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.602</td>
<td>0.646</td>
<td>0.600</td>
<td>0.589</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.150</td>
<td>0.199</td>
<td>0.170</td>
<td>0.176</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.670</strong></td>
<td><strong>0.793</strong></td>
<td><strong>0.679</strong></td>
<td><strong>0.668</strong></td>
</tr>
<tr>
<td>Pair tri-gram baseline</td>
<td>0.590</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>PHMM0</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Viterbi</td>
<td>0.521</td>
<td>0.478</td>
<td>0.588</td>
<td>0.590</td>
</tr>
<tr>
<td>Forward</td>
<td>0.628</td>
<td>0.595</td>
<td>0.637</td>
<td>0.635</td>
</tr>
<tr>
<td>Vit. log-odds (identical)</td>
<td>0.250</td>
<td>0.280</td>
<td>0.255</td>
<td>0.261</td>
</tr>
<tr>
<td>Vit. log-odds (distinct)</td>
<td>0.709</td>
<td>0.748</td>
<td>0.698</td>
<td>0.693</td>
</tr>
<tr>
<td>Forward log-odds (identical)</td>
<td>0.261</td>
<td>0.325</td>
<td>0.282</td>
<td>0.285</td>
</tr>
<tr>
<td>Forward log-odds (distinct)</td>
<td><strong>0.772</strong></td>
<td><strong>0.865</strong></td>
<td><strong>0.774</strong></td>
<td><strong>0.765</strong></td>
</tr>
<tr>
<td>Pair tri-gram baseline</td>
<td>0.659</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.17: English-Thai Pair HMM transliteration identification accuracy and MRR

transliteration identification quality for all language pairs. We also see that although there is a general improvement in accuracy and MRR from the lack of Pair HMM transition parameters (PHMM0) to using the transition parameters (PHMM3, PHMM5, PHMM9), an increase in the number of transition parameters does not guarantee improvements in transliteration identification quality. This is the case with PHMM9 which despite having the best accuracy and MRR on the Forward and Viterbi algorithm for almost all language pairs, results in lower transliteration identification (TI) quality on the best performing Forward log-odds algorithm. It is also surprising to see that for at least four of the language pairs (En-Ba, En-Ch, En-Hi, and En-Th), the Forward log-odds algorithm for PHMM3 results in the best overall accuracy and MRR while the other algorithms lead to poor TI quality in comparison to other Pair HMM variants. For the English-Russian dataset, we expected much better transliteration identification quality from the the Pair HMMs since Cyrillic and Latin alphabets that are used in Russian and English respectively are both mostly phonemic and therefore should simplify the application of the Pair HMMs directly to the original orthography; however, as Table 4.15 shows, none of the Pair HMMs results in an accuracy or MRR higher than that for the baseline Pair trigram and higher order n-grams (Table 4.10). We applied the Baum-Welch algorithm for each of the Pair HMM variants on the English-Russian datasets this time with random
probability distributions for the initial parameters in order to verify the results in Table 4.15, but there was no change in transliteration identification accuracy and MRR. The English-Russian dataset is the only dataset where the Pair HMMs have resulted in considerably lower TI accuracy and MRR compared to the case of the baseline approach using Pair tri-gram and higher order n-gram relationships.

4.6 Conclusion

In this chapter, we have evaluated several Pair HMM settings in an experimental transliteration identification task. In the first set of experiments where we evaluate the models in identifying transliterations on a geographic names corpora, results show that it is important to use models that accurately capture the properties of character representations in different writing systems. A comparison of the transliteration similarity scoring algorithms shows that it is important to factor in a measure for the random probability of relating two candidate name pairs. We see that the Forward log-odds algorithm consistently leads to the best transliteration identification accuracy and MRR. We also see that the use of a log-odds score always results in better transliteration identification quality across all model settings. The results also suggest the insensitivity of using the log-odds similarity score on changes in model parameterizations. For example, we see that the Pair HMM variant that uses only 3 transition parameters and a Forward log-odds algorithm leads to the best overall performance despite a poor performance by the other Pair HMM algorithms using this model. An investigation into changes in Pair HMM state transition parameters shows that it is important to use transition parameters in the computation of transliteration similarity. However, the results also suggest that an increase in transition parameters does not guarantee an improvement in transliteration identification quality.
5.1 Introduction

So far, we have adapted the approach of Pair Hidden Markov models (Pair HMMs) to compute transliteration similarity with promising results in an experimental transliteration identification task. In this chapter, we introduce the second DBN-related approach that we adapt to estimate transliteration similarity. This approach is based on the representation of an edit-distance stochastic transducer as an edit-distance based Dynamic Bayesian Network (DBN) model (Filali and Bilmes 2005). Unlike the Pair HMM approach whose implementation is based on the classic Hidden Markov model (HMM) framework, the edit-distance based DBN approach is implemented as a template-based approach. In using the edit-distance based DBN approach, we can define several DBN model generalizations while addressing some restrictions that are fundamental to HMMs for example with regard to the Markovian assumptions of conditional independence and the definition of an HMM as a function of a single independent variable. The edit-distance based DBN addresses some of the HMM limitations by allowing the definition of an arbitrary number of random variables and dependencies between the variables while modeling various factors that we may hypothesize to have an effect in the computation of transliteration similarity.

In the next section, we review the origins of the edit-distance based DBN approach. Later, we describe different DBN model generalizations that we have adapted to estimate transliteration similarity. We then report on the results associated with the application of models from the DBN model generalizations in an experimental
transliteration task with a comparison to the results from the application of the Pair HMMs and pair n-gram model approaches.

5. Transduction-based DBN models

The Transduction-based DBN approach (Filali and Bilmes 2005) is founded on the representation and implementation of a memoryless stochastic transducer (initially proposed by Ristad and Yianilos (1997)) as a DBN model for learning string edit distance. It is helpful to review the formulation of the memoryless stochastic transducer so as to understand the definitions of the random variables and the conditional probability distributions in the construction of the corresponding DBN model.

5.2 The memoryless stochastic transducer

Ristad and Yianilos (1997) model string edit distance as a memoryless stochastic transduction between two strings. The transductions involve any of the three edit operations of generating a substitution pair of symbols denoted by \( \langle a, b \rangle \), a deletion pair \( \langle a, \epsilon \rangle \), an insertion pair \( \langle \epsilon, b \rangle \) or a termination. In order to compute the probability for a pair of strings, Ristad and Yianilos consider a string pair to be an equivalence class representative for all edit sequences whose yield is that string pair. The probability of a string pair is therefore computed as the sum of the probabilities of all edit sequences for that string pair.

Assuming that the source string is denoted by \( s_1^m = s_1 s_2 ... s_m \) where the characters are associated with the source alphabet \( A_s \), and the target string is denoted by \( t_1^n = t_1 t_2 ... t_n \) for a target alphabet \( A_t \). The edit operations can be represented using a hidden random variable \( Z \) that takes in values from \( (A_s \cup \epsilon \times A_t \cup \epsilon) \setminus (\epsilon, \epsilon) \). Following Ristad and Yianilos’ consideration in the last paragraph, \( Z \) can be perceived as a random vector with two components \( (Z(s), Z(t)) \). The probability of a pair of strings can therefore be computed by marginalizing over the two components of the edit operation variable \( Z \) (Filali and Bilmes 2005):

\[
P(s_1^m, t_1^n | \theta) = \sum_{z(1)} \sum_{z(t)} P(z_1^l, s_1^m, t_1^n | \theta) \tag{5.1}
\]

where \( \theta \) represents the parameters for the memoryless stochastic transducer and \( z_1^l \) is such that its yield denoted by \( v(z_1^l) = (s_1^m, t_1^n) \) and \( \max(m, n) \leq l \leq m + n \).

In the memoryless stochastic transducer, there is no dependence between edit operations. Therefore \( P(z_1^l, s_1^m, t_1^n | \theta) \) is simply the product of the probabilities of the individual edit operations, that is
5.2 Transduction-based DBN models

\[ P(z^i, s^m_i, t^m_i \mid \theta) = \prod_i P(z^i, s^m_i, t^m_i \mid \theta) \text{ for } 1 \leq i \leq l \text{ and } z_i = \langle z^s_i, z^t_i \rangle \]

The RY memoryless stochastic transducer is also context-independent in the sense that the edit operation random variable \( z_i \) does not have any local dependence on the source and target string characters \( s_a \) and \( t_b \) respectively, but does have a global context dependence that ensures the generation of the string pair \( (s_m, t_n) \).

Using \( Q(z^i) \) to represent \( P(z^i, s^m_i, t^m_i \mid \theta) \), this local context independence is depicted in the following expression (Filali and Bilmes 2005):

\[
Q(z^i) \propto \begin{cases} 
  f_{\text{ins}}(t_b^i) & \text{for } z^s_i = \epsilon; \ z^t_i = t_b^i \\
  f_{\text{del}}(s_a^i) & \text{for } z^s_i = s_a^i; \ z^t_i = \epsilon \\
  f_{\text{sub}}(s_a^i, t_b^i) & \text{for } (z^s_i, z^t_i) = (s_a^i, t_b^i) \\
  0 & \text{otherwise} 
\end{cases} \tag{5.2}
\]

where \( \sum z Q(z) = 1 \); \( a_i = \sum_{j=1}^{i-1} 1_{\{z^s_j \neq \epsilon\}} \) and \( b_i = \sum_{j=1}^{i-1} 1_{\{z^t_j \neq \epsilon\}} \) are the respective indexes of the source and target string generated up to the \( i^{th} \) edit operation; and \( f_{\text{ins}}, f_{\text{del}}, \) and \( f_{\text{sub}} \) are the functions mapping to \([0,1]\). Equation 5.2 also enforces the consistency constraint that the pair of characters output by \( (z^s_i, z^t_i) \) is the same as the actual pair of characters \( (s_a^i, t_b^i) \) that needs to be generated at the \( i^{th} \) transduction so that the total yield, \( v(z^l) \) is equal to the string pair \( (s^m_i, t^m_i) \).

5.2.2 Representing the RY transducer as a DBN

Filali and Bilmes (2005) use the graphical models framework to represent the joint probability distribution of the Ristad-Yianilos (RY) stochastic transducer model with its related consistency constraints as a DBN model. Random variables are defined for the DBN model to correspond to the objects that contribute to the edit distance computation in the memoryless stochastic transducer. With reference to the previous section, the main objects of interest include: the edit operation variable \( Z_i \); source and target character variables; variables that capture the position of the characters in the source and target strings; and consistency variables that check the yield of the edit operation variable against the actual pair of characters at a given position as defined above. The dependencies between the random variables follow naturally. For example, a dependency is defined between a string position variable and the character variable. In this case, the idea is that knowledge about the position in a string leads to knowledge about the character at that position. For consistency checking, the consistency variables are defined to depend on the character variables and the edit operation variable. To complete the specification of the DBN model, we need to define the initial Bayesian network \( (B_0) \), and a 2-frame Temporal Bayes Net
5. Transduction-based DBN models for machine transliteration

(2-TBN) ($\mathcal{B}_{\infty}$). We use the same graphical modeling approach in Filali and Bilmes (2005) where the main modeling terms are associated with concepts in Automatic Speech Recognition (ASR).

In the ASR-based graphical modeling approach, a frame$^1$ is used to represent a set of random variables and their attributes at a given time. The term Prologue frame is used to refer to the initial Bayesian network $\mathcal{B}_0$, and the term Chunk frame is used to refer to the Bayesian network that is to be unrolled as defined for a 2-TBN. An Epilogue frame refers to a Bayesian network that models end conditions. Using this ASR-based notation, the initial, chunk, and epilogue Bayesian networks that model the memoryless stochastic transducer are as shown in Figure 5.1.

Since the networks are designed to model the properties of the memoryless stochastic transducer, the DBN template defined by the three networks in Figure 5.1 is also referred to as memoryless and context-independent (MCI). In the following paragraph, we describe the variables and dependencies used to specify the MCI DBN template in Figure 5.1.

The nodes in Figure 5.1 represent variables while the directed edges represent ‘informational relationships’ between the variables. As defined in chapter 3, the directed edges are used to suggest the influence of one variable on another. In Figure 5.1 $Z$ as defined above is the edit operation variable and is defined for the DBN model to take values from a distribution whose cardinality is equal to the product of the size of the source ($n_{A_s \cup \epsilon}$) and target ($n_{A_t \cup \epsilon}$) vocabularies excluding $\epsilon$. $sp$ and $tp$ are variables used to capture the current position in the source and target strings respectively. $sp$ has a cardinality ($m$) equal to the length of the source string while $tp$ has a cardinality ($n$) equal to the length of the target string. The $s$ and $t$ variables represent the current character in the source and target strings respectively. The cardinality of $s$ is equal to the size of the source language alphabet including the empty string ($A_s \cup \epsilon$) while that for $t$ is equal to the size of the target language alphabet ($A_t \cup \epsilon$). The $sc$ and $tc$ nodes enforce the consistency constraint$^2$ implied by equation 5.2. The send node is introduced for the DBN model as a ‘switching’ parent of $Z$ and it is used to indicate when we are past the end of both the source and target strings, that is, when $sp > m$ and $tp > n$. The send and tend nodes represent variables that ensure that we are past the end of the source and target strings respectively.

---

$^1$The term “frame” in the context of ASR, refers to contiguous, small regions of a speech signal which aid in the identification of phonemes

$^2$Following equation 5.2, the $sc$ and $tc$ nodes have a fixed observed value 1 and the only configuration of their parents is such that the source component of the edit operation variable $Z$ is $s$ or an empty symbol for $sc$ and the target component of the edit operation $Z$ is $t$ or an empty symbol for $tc$ and that $Z$ does not generate empty source and target symbols at the same time (Filali and Bilmes 2005).
5.2 Transduction-based DBN models

Decision trees are used to implement deterministic conditional probability tables for most of the dependencies in Figure 5.1. Specifically, decision trees are used for each of the following variable dependencies: $sp_i \rightarrow s_i$ and $tp_i \rightarrow t$ (for the relationship between position and character variables for the source and target strings respectively); $s_i \rightarrow sc_i \leftarrow Z_i$ and $Z_i \rightarrow tc_i \leftarrow t_i$ (for the relationship between the consistency variables and their respective parents); $sp_{i-1} \rightarrow sp_i \leftarrow Z_{i-1}$ and $tp_{i-1} \rightarrow tp_i \leftarrow Z_{i-1}$ (for transition on the source and target side respectively); $spos_i \rightarrow end_i \leftarrow tpos_i$ (for determining whether we are past the end of the edit sequence); $end_i \rightarrow Z_i$ (for determining a value that $Z_i$ should have depending on the value (0 or 1) of $end_i$, that is $end_i$ is a switching parent of $Z_i$, when $end_i = 0$, the
conditional probability distribution of \( Z_i \) is as described above with a cardinality of \( n_{A_A \cup \epsilon} \times n_{A_A \cup \epsilon} \), and when \( \text{end}_i = 1 \), \( Z_i \) takes with a probability 1, a fixed value outside the range of edit operations but consistent with \( s_i \) and \( t_i \); \( sp_i \rightarrow send_i \leftarrow Z_i \) and \( t_p \rightarrow tend_i \leftarrow Z_i \) (for determining whether we are at or past the end of the source and target strings respectively. This is not the same as for the \( \text{end}_i \) variable where we only check whether we are past the end of the edit sequence. So if \( sp_i > m \) (the length of the source string), then \( send_i \) is observed to be 1 with a probability 1; else if \( send_i < m \), then \( P(send_i = 1) = 0 \) and the whole edit sequence is considered to have zero probability; else if \( sp_i = m \), then \( send_i \) will have a probability greater than 1 only if the \( Z_i \) is an insertion). For the \( Z_i \) variable, a Dense Probability Mass Function (DPMF) is used to implement a Dense Conditional Probability Table (DCPT) that it uses to generate the source and target symbols in each time slice.

5.2.3 DBN templates for modeling transliteration similarity

In order to propose additional edit distance based DBN templates using the Filali and Bilmes’ (2005) approach, we have to use the MCI DBN template as the baseline template. Our starting point would be to represent the classic HMMs which are the simplest DBN models. In this section, we discuss the requirements for representing Pair HMMs as edit distance based DBN models based on the MCI DBN template.

We begin by using an alignment example to help clarify the difference between the Pair HMMs as introduced in Chapter 4 and an edit distance based DBN representation. Consider an arbitrary one-to-one character alignment as illustrated in Table 5.1 between the Russian name ‘Анатольевич’ and its Dutch representation ‘Anatoljevitsj’. The upper part of Figure 5.2 is a finite state representation of a Pair HMM for the alignment while the lower diagram is a corresponding conceptual DBN representation. A comparison of the two diagrams in Figure 5.2 shows at least two differences. One difference is that nodes and arcs in the Pair HMM representation refer to states and transitions respectively, while in the DBN representation, nodes represent variables and arcs specify conditional dependencies / independencies between the variables. The other difference is that the labeling of the hidden variable \( (Z_i) \) shows that the DBN representation is explicit about time.

Table 5.1: An arbitrary alignment between Russian word Анатольевич and Dutch representation Anatoljevitsj.
5.2 Transduction-based DBN models

Figure 5.2: A Pair HMM (top) and a conceptual DBN (bottom) representation for the alignment in Table 5.1. Shaded nodes in the DBN representation indicate ‘observed’ variables and unshaded ones indicate ‘hidden’ variables. \( \epsilon \) denotes the ‘empty’ symbol.

Figure 5.3: A DBN representation of a Pair HMM with an illustrative assignment of values to the variables. Light gray nodes are used here to illustrate partially deterministic variables. Dark gray nodes represent deterministic variables while unshaded nodes represent hidden variables.

While the DBN representation in Figure 5.2 seems to capture the Pair HMM representation, the DBN representation does not yet represent the states of the Pair HMM at each time frame. If we follow the graphical modeling approach in the previous section, we need to introduce the position variables in a manner similar to that as described for the MCI DBN model. The position variables will enable the
representation of a Pair HMM at particular time frames. Figure 5.3 shows the DBN representation of the Pair HMM above with the position variables added; if we have a substitution operation, the source and target position variables are incremented; if we have an insertion, the source position variable will retain its previous value while the target position variable is incremented; and if we have a deletion, the target position variable retains its previous value while the source position variable is incremented. We can specify the \( Z \) variable to model the three Pair HMM emission states by taking values from a distribution with a cardinality of \( n_{A_s} \times n_{A_t} + 1 \). The Pair HMM transition probabilities can be encoded in the transition variables (that is between the edit operation variable \( Z_{i-1} \) and the position variables \( sp_i \) and \( tp_i \)). We could then employ decision trees to implement the CPTs of the different dependencies in a similar way as was described for the MCI DBN model.

Based on the MCI DBN template, we adapt three other DBN model templates that were initially introduced in (Filali and Bilmes 2005) to compute transliteration similarity. The DBN model templates represent different dependencies on the edit operation random variables including: edit operation memory dependencies; source and/or target character context dependencies; and edit operation length dependencies. In the following, we briefly point out the main properties of these other DBN models.

a) Context-independent memory DBN template

Figure 5.4 shows the initial Bayesian network and the 2-TBN for the context-independent memory (MEM) DBN model. Here, modeling memory, means having the current edit operation variable \( Z_i \) use information from the previous edit operation variable \( Z_{i-1} \). In Figure 5.4, a variable \( H \) is introduced to model various dependencies between \( Z_{i-1} \) and \( Z_i \). Generally, \( H \) can be stochastic or deterministic, and the amount of information that it summarizes from one frame to another is determined by its cardinality (Filali and Bilmes 2005). In this chapter, we only investigate the deterministic implementation of \( H \), where \( H \) allows the modeling of the conditional probability distribution \( P(Z_i|Z_{i-1}) \). Apart from this type of dependency, \( Z_i \) can also depend on the type of edit operation in the previous frame.

b) Context-dependent DBN template

Figure 5.5 shows the initial Bayesian network and the 2-TBN for the context-dependent DBN template. Context-dependence, here, means adding a dependence of the edit operation variable on the source and/or target string characters. For example, as shown in Figure 5.5, we model context-dependence of the edit operation on the source string characters by adding edges from \( s_i \), \( sprev_i \) to \( Z_i \). As shown in Figure
5.2 Transduction-based DBN models

5.5, the source consistency variable $sc$ is not used since the consistency constraint is defined through the conditional probability table of $Z$ given its parents. Generally, we can model context dependence to range from the case where we include only the dependence on the current character to the case where we consider all characters in the source (and/or target) string. In this chapter, we investigate two main cases: one case as shown in Figure 5.5 and the other case where we consider only the current character in the source string. Based on these two cases, we investigate context-dependence for two additional settings where we simply change the datasets so that the source becomes target and target becomes source.

Figure 5.4: Graphical representation for the context-independent memory DBN template. (Adapted from Filali and Bilmes, 2005).
c) Length DBN template

Figure 5.6 shows the initial Bayesian network and the 2-TBN for a context-dependent length DBN template. To enable the investigation of the effect of the length of the edit sequence on a string similarity estimate, additional variables which model the logic necessary for simulating variable length-unrolling of the chunk frame are introduced. These additional variables in Figure 5.6 are defined as follows. incl is a stochastic hidden random variable whose value added to that of the variable inilen determines the number of allowed edit operations. The variable cnt is used to determine the index of the current edit operation and is used to trigger the random variable reql when the required sequence length is reached. The variable end is explained if we reach the end of one string after having reached the end of the other string in a previous frame. Apart from the template in Figure 5.6, we also investigate the effect of length for a context-independent DBN model by simply adding similar edges introduced by the additional variables in this section to the MCI DBN template in Figure 5.1.

Figure 5.5: Graphical representation for the context-dependent DBN template.
5.2 Transduction-based DBN models

Figure 5.6: Graphical representation for a context-dependent length DBN template.

d) Transliteration data requirements

The application of the DBN model templates above in the specification of DBN models to compute transliteration similarity requires the representation of transliteration data in a form suitable for running DBN inference algorithms. Most of the requirements which we discussed for the application of the Pair HMMs to compute transliteration similarity apply for the DBN templates introduced in this chapter. We follow the same segmentation approach as discussed in Chapter 4. That is we tokenize
source and target strings per character. We also assume a monotonic ordering of the strings, and a one-to-one correspondence between tokens. The limitations described for the Pair HMMs based on these assumptions also apply to the DBN templates. In this chapter, we specify the DBN templates so that the resulting DBN models are based on the representation of separate source and target alphabets corresponding the respective writing systems. However, we do not investigate the effect of using separate alphabets against the use of one alphabet on the quality of transliteration similarity estimates as we did in Chapter 4.

5.2.4 Inference

We define the DBN templates for computing transliteration similarity using the Graphical Modeling Toolkit (GMTK) (Bilmes and Zweig 2002). Our interest is in two main inference tasks when applying the edit-distance based DBN templates: parameter estimation given training transliteration data, and using the trained DBN models to compute transliteration similarity. In both cases, GMTK uses the frontier algorithm which we describe in the following.

a) Frontier algorithm

The frontier algorithm (Zweig 1996) uses a Forwards-Backwards procedure which updates the joint distribution over a set of hidden nodes without needing to create and manipulate a huge transition matrix as is the case in the Forwards-Backwards algorithm for the classic HMMs. The forwards-backwards procedure in both cases assumes that a hidden node \( Z_t \) or set of hidden nodes \( Z^{(1:D)}_t \) at a given time \( t \) d-separates the past from the future. For the DBNs, the Frontier algorithm uses a Markov blanket over the hidden nodes which it “sweeps” across the DBN first in the forwards direction (while computing the forward variable \( \alpha \)) and then backwards (for computing the backward variable \( \beta \)) (Murphy 2002). Our review of the Frontier algorithm follows from a simplified presentation of the same by Murphy (2002). We also use the same notation in (Zweig 1996).

Forwards pass

The nodes in the Markov blanket are called the “Frontier set” and the set is denoted by \( F \); the nodes to the left and right of \( F \) are denoted by \( L \) and \( R \). At every step of the frontier algorithm, \( F \) should d-separate \( L \) and \( R \).

Let \( h_F \) refer to the hidden nodes in \( F \), \( e_F \) to the evidence nodes in \( F \), \( e_L \) to the evidence nodes in \( L \), and \( e_R \) to the evidence nodes in \( R \). In the forward pass, the
5.2 Transduction-based DBN models

probability of the nodes in $F$ is expressed as $P(F) \overset{\text{def}}{=} P(h_F, e_F, e_L)$, and is computed recursively as follows.

We can add a node $N$ to the frontier set (that is move it from $R$ to $F$) when all its parents are already in $F$:

$$P(e_L, e_F, h_F, N) = P(e_L, e_F, h_F)p(N|e_F, h_F)$$

(5.3)

since $N$ is conditionally dependent on only $e_F$ and $h_F$ but not on $e_L$. Equation 5.3 means that adding a node consists of multiplying its conditional probability distribution (CPD) into the frontier.

We can remove a node $N$ from $F$ to $L$ when all its children are in $F$. If $N$ is hidden, then $e_{L \cup \{N\}} = e_L$ and $e_{F \setminus \{N\}} = e_F$.

$$P(e_L, e_F, h_F, N) = P(e_L, e_F, h_F)$$

which means that removing a node consists of marginalizing it out. The same applies to the case where $N$ is observed:

$$P(e_L, e_F, h_F, N) = P(e_L, e_F, h_F)$$

Backwards Pass

In the backwards pass, $P(F) \overset{\text{def}}{=} P(e_R|h_F, e_F)$. The frontier is advanced from frame $t + 1$ to frame $t$ by adding and removing nodes in the opposite order that is used in the forwards pass. Adding a node in this case means moving it from $L$ to $F$, and removing a node means moving it from $F$ to $R$.

When a node $N$ is added to $F$, we compute $P(e_R|e_F, h_F, N)$. In this case, since $N$’s children are in $F$, which “shield” $N$ from $e_R$: $P(e_R|e_F, h_F, N) = P(e_R|e_F, h_F)$. When a node $N$ is removed from $F$, and added to $R$, we compute $P(e_{R \cup \{N\}}|e_{F \setminus \{N\}}, h_{F \setminus \{N\}})$ from $P(e_R|e_F, h_F)$. If $N$ is hidden, then $e_{R \cup \{N\}} = e_R$, and $e_{F \setminus \{N\}} = e_F$. Therefore

$$P(e_{R \cup \{N\}}|e_{F \setminus \{N\}}, h_{F \setminus \{N\}}) = P(e_R|e_F, h_{F \setminus \{N\}})$$
\[= \sum_{N} P(N, e_R|e_F, h_{F\setminus(N)}) \]
\[= \sum_{N} P(N|e_F, h_{F\setminus(N)}) P(e_R|N, e_F, h_{F\setminus(N)}) \]
\[= \sum_{N} P(N|e_F, h_{F\setminus(N)}) P(e_R|e_F, h_F) \]

This means that to remove a node \( N \), we multiply in \( N \)'s CPD.

The same procedure applies to the case where \( N \) is observed only that there is no need to marginalize out \( N \) since it has only one possible value.

\[ P(e_R\cup(N)|e_F\setminus(N), h_{F\setminus(N)}) = P(e_R\cup(N)|e_F\setminus(N),h_F) \]
\[= P(e_N, e_R|e_F\setminus(N), h_F) \]
\[= P(e_N|e_F\setminus(N), h_F) P(e_R|e_N, e_F\setminus(N), h_F) \]

b) Generalized Expectation Maximization

In the transliteration identification experiments, we use GMTK’s implementation of the generalized maximization (GEM) algorithm to estimate parameters for each of the transduction-based DBN models. The reader is referred to Chapter 3 for a detailed explanation of Expectation Maximization (EM) and the case for GEM. The parameters for the DBN models constitute conditional probability tables that encode the relationships between variables in a frame and between two adjacent frames, and values for decision trees that are used to represent deterministic mappings. Specifically, there are two types of parameters (Bilmes 2002): numerical parameters which in our case include dense and sparse CPTs and for which the EM algorithm is used; and non-numerical parameters, which in our case constitute decision tree deterministic values that do not require the use of the EM algorithm.

5.3 Experiments (NEWS 2009 and 2010 shared task data)

5.3.1 Data

To ensure a comparison of the edit distance based DBN models introduced in this chapter and the Pair HMMs, we use transliteration data from the NEWS 2009 and 2010 shared tasks on transliteration generation. This transliteration data for the seven language pairs has already been described in Chapter 4, section 4.5.1.
5.3 Experiments (NEWS 2009 and 2010 shared task data)

5.3.2 Evaluation setup and results

In using the NEWS 2009 and 2010 shared task data, we follow the same transliteration identification setup described in Chapter 4, section 4.5.2. We train the DBN models on the same training data (for each language pair) that we used for training the Pair HMMs and pair n-gram models. In estimating the parameters for each DBN model, we applied the generalized EM algorithm with a maximum specification of three iterations. During pre-runs, we observed this number of iterations to be optimal in avoiding the overfitting of the models. We then applied a scoring algorithm implemented in the GMTK toolkit to estimate transliteration similarity using the parameters of the trained DBN models. The DBN models are evaluated on transliteration identification accuracy and mean reciprocal rank as defined in Chapter 4. Table 5.2 shows the transliteration identification accuracy and MRR results for the transduction-based DBN models as compared to the best performing Pair HMMs (Phmm) and the best baseline results for the Pair tri-gram method on the same seven transliteration datasets. In Table 5.2: MCI refers to the memoryless and context-independent DBN model (Figure 5.1), MEM refers to the memory-dependent DBN model (Figure 5.4), CONs1 and CONs2 represent DBN models where the dependence of the edit operation on the characters in the source string is factored into the string similarity estimate, while CONt1 and CONt2 represent DBN models where the dependence of the edit operation on the target string character(s) is used. For CONs1, the edit operation depends on the current character in a source string while for CONs2, the edit operation depends on both the current and previous character in the source string (Figure 5.5). CONt1 and CONt2 correspond respectively to CONs1 and CONs2, but with t now referring to the target character(s).

Table 5.2 shows that the edit distance based DBN models improve transliteration identification accuracy and MRR when applied to transliteration data for five of the seven language pairs (En-Ba, En-Ch, En-Ru, En-Ta, and En-Th). On the En-Ka dataset, the best accuracy and MRR from the DBN models (CONt2 and CONs2 respectively) is the same as that for the best Pair HMM. Unlike the case for the Pair HMMs, the DBN here models have much better accuracy and MRR compared to the baseline pair n-gram approach for all language pairs including the case for English-Russian. The memory-dependent DBN model results in the lowest accuracy and MRR compared to all the other DBN models which is quite surprising especially when compared with the MCI DBN model where no edit operation dependencies are used nor character context dependencies. The relatively good results from the context-dependent DBN models underline the necessity to represent character context dependencies in the edit operation random variable.
Table 5.2: Transliteration identification accuracy and MRR for different models

<table>
<thead>
<tr>
<th>Models</th>
<th>En-Ba</th>
<th>En-Ch</th>
<th>En-Hi</th>
<th>En-Ka</th>
<th>En-Ru</th>
<th>En-Ta</th>
<th>En-Th</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCI</td>
<td>0.868</td>
<td>0.299</td>
<td>0.747</td>
<td>0.718</td>
<td>0.978</td>
<td>0.646</td>
<td>0.351</td>
</tr>
<tr>
<td>MEM</td>
<td>0.894</td>
<td>0.487</td>
<td>0.717</td>
<td>0.576</td>
<td>0.888</td>
<td>0.717</td>
<td>0.489</td>
</tr>
<tr>
<td>CONs1</td>
<td>0.960</td>
<td>0.699</td>
<td>0.855</td>
<td>0.842</td>
<td>0.983</td>
<td>0.828</td>
<td>0.736</td>
</tr>
<tr>
<td>CONs2</td>
<td>0.957</td>
<td>0.796</td>
<td>0.857</td>
<td>0.845</td>
<td>0.974</td>
<td>0.853</td>
<td>0.789</td>
</tr>
<tr>
<td>CONt1</td>
<td>0.954</td>
<td>0.676</td>
<td>0.840</td>
<td>0.835</td>
<td>0.983</td>
<td>0.840</td>
<td>0.762</td>
</tr>
<tr>
<td>CONt2</td>
<td>0.960</td>
<td>0.820</td>
<td>0.859</td>
<td>0.856</td>
<td>0.976</td>
<td>0.862</td>
<td>0.848</td>
</tr>
<tr>
<td>LENs1</td>
<td>0.956</td>
<td>0.713</td>
<td>0.831</td>
<td>0.767</td>
<td>0.982</td>
<td>0.839</td>
<td>0.733</td>
</tr>
<tr>
<td>LENt1</td>
<td>0.954</td>
<td>0.698</td>
<td>0.847</td>
<td>0.806</td>
<td>0.976</td>
<td>0.797</td>
<td>0.700</td>
</tr>
<tr>
<td>best PHMM</td>
<td>0.932</td>
<td>0.684</td>
<td>0.885</td>
<td>0.858</td>
<td>0.888</td>
<td>0.827</td>
<td>0.793</td>
</tr>
<tr>
<td>pair tri-gram</td>
<td>0.804</td>
<td>0.524</td>
<td>0.813</td>
<td>0.763</td>
<td>0.944</td>
<td>0.775</td>
<td>0.623</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MRR</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MCI</td>
<td>0.910</td>
<td>0.423</td>
<td>0.827</td>
<td>0.806</td>
<td>0.985</td>
<td>0.751</td>
<td>0.479</td>
</tr>
<tr>
<td>MEM</td>
<td>0.914</td>
<td>0.623</td>
<td>0.774</td>
<td>0.634</td>
<td>0.921</td>
<td>0.809</td>
<td>0.612</td>
</tr>
<tr>
<td>CONs1</td>
<td>0.975</td>
<td>0.791</td>
<td>0.908</td>
<td>0.897</td>
<td>0.986</td>
<td>0.896</td>
<td>0.833</td>
</tr>
<tr>
<td>CONs2</td>
<td>0.972</td>
<td>0.869</td>
<td>0.908</td>
<td>0.901</td>
<td>0.979</td>
<td>0.913</td>
<td>0.868</td>
</tr>
<tr>
<td>CONt1</td>
<td>0.972</td>
<td>0.775</td>
<td>0.897</td>
<td>0.895</td>
<td>0.987</td>
<td>0.905</td>
<td>0.849</td>
</tr>
<tr>
<td>CONt2</td>
<td>0.975</td>
<td>0.887</td>
<td>0.908</td>
<td>0.908</td>
<td>0.981</td>
<td>0.920</td>
<td>0.913</td>
</tr>
<tr>
<td>LENs1</td>
<td>0.971</td>
<td>0.807</td>
<td>0.888</td>
<td>0.839</td>
<td>0.985</td>
<td>0.903</td>
<td>0.828</td>
</tr>
<tr>
<td>LENt1</td>
<td>0.970</td>
<td>0.786</td>
<td>0.901</td>
<td>0.871</td>
<td>0.981</td>
<td>0.867</td>
<td>0.791</td>
</tr>
<tr>
<td>best PHMM</td>
<td>0.950</td>
<td>0.777</td>
<td>0.923</td>
<td>0.905</td>
<td>0.892</td>
<td>0.886</td>
<td>0.865</td>
</tr>
<tr>
<td>pair tri-gram</td>
<td>0.844</td>
<td>0.604</td>
<td>0.850</td>
<td>0.806</td>
<td>0.956</td>
<td>0.824</td>
<td>0.685</td>
</tr>
</tbody>
</table>

The results for the DBN models in table 5.2 are quite similar to those reported for the pronunciation classification task in (Filali and Bilmes 2005) where the context-dependent related models resulted in the lowest word error rate. However, since we have applied the DBN models on more than one language pair, we see some variations in transliteration identification quality. The CONs1 and CONt2 models perform better than the the other context-dependent DBN models on the English-Bangla dataset while the CONs1 performs best on the English-Russian dataset, and CONt2 performs best on the remaining datasets. In (Filali and Bilmes 2005), CONs2 had the best result from their set of context-dependent DBN models. Our results also suggest that the language where we model context can have a considerable effect on identification accuracy. If we compare the results for CONs1 and CONt1, and those for CONs2 and CONt2, we see that the modeling of context on the target language.
5.3 Experiments (NEWS 2009 and 2010 shared task data)

The results in Table 5.2 clearly show that we still need to improve transliteration identification quality for most of the language pairs. An analysis of sample results for the same input could be helpful in finding ways of achieving higher transliteration identification quality based on the application of DBN models. Table 5.3 shows a sample of the English-Thai results for 16 test names where at least one of the context-dependent models did not return the correct candidate at first rank. From the sample results in the table alone, we can see that although all DBN models are context-dependent, they return different results at the first rank. We also see that the reference NE is returned at different ranks. The results in the table also suggest that if we had a way to combine the models so that the system used only the best result, we would improve transliteration identification accuracy compared to the individual use of the models. However, it would be difficult to model this decision-making process since we do not know the similarity estimate that would result in the highest rank for the models. But we can still explore other simple schemes of combining the models to determine whether there can be improvements in transliteration identification quality. In next subsection we investigate the different combinations of the context-dependent DBN models to compute transliteration similarity.

5.3.4 Computing transliteration similarity based on ensembles of DBN models

The ensemble framework we follow in investigating the combined application of context-dependent DBN models to compute transliteration similarity is quite straightforward. First of all, we estimate the parameters of each context-dependent DBN model separately. We then apply the trained models separately to compute transliteration similarity estimates for the candidate string pairs. We then combine the individual model similarity estimates by adding them for each candidate string pair and getting an average based on the total number of models.

Table 5.4 shows the transliteration identification results where the transliteration similarity estimate from each model is simply added to that of the other model(s) to obtain a new similarity estimate which is then used for ranking. We have restricted the combinations to the context-dependent DBN models since they have highest accuracy and MRR results compared to the other DBN models. In Table 5.4, CONs1, CONs2, CONt1, CONt2 are as described above while ALLCONst represents the
<table>
<thead>
<tr>
<th>Src. NE</th>
<th>Ref. NE</th>
<th>CONs1</th>
<th>CONs2</th>
<th>CONt1</th>
<th>CONt2</th>
</tr>
</thead>
<tbody>
<tr>
<td>holeman</td>
<td>โฮล/uni0E4C lowเมิ่น</td>
<td>ฮัลต/uni0E4C lowเมิ่น</td>
<td>โฮลส/uni0E4C lowเมิ่น</td>
<td>holzman(2)</td>
<td>holzman(4)</td>
</tr>
<tr>
<td>rickett</td>
<td>ริคต</td>
<td>ริคโค</td>
<td>ริกส/uni0E4C low</td>
<td>hultman(3)</td>
<td>hultman(3)</td>
</tr>
<tr>
<td>dare</td>
<td>แดร/uni0E4C lowก</td>
<td>แดร/uni0E4C lowต</td>
<td>เอลเดอร/uni0E4C low</td>
<td>dark(6)</td>
<td>dart(8)</td>
</tr>
<tr>
<td>deibel</td>
<td>ไดเบิล</td>
<td>เอลเดอร</td>
<td>abell(4)</td>
<td>holzman(2)</td>
<td>holzman(2)</td>
</tr>
<tr>
<td>guide</td>
<td>ไกด/uni0E4C low</td>
<td>กิโด้/uni0E4C low</td>
<td>เกลด/uni0E4C low</td>
<td>reco(22)</td>
<td>geld(6)</td>
</tr>
<tr>
<td>wright</td>
<td>ไรต</td>
<td>แรรง</td>
<td>Wareing(7)</td>
<td>reiki(11)</td>
<td>reiki(11)</td>
</tr>
<tr>
<td>mier</td>
<td>มี่เออร</td>
<td>เมเออร</td>
<td>abell(4)</td>
<td>moore(4)</td>
<td>moore(4)</td>
</tr>
<tr>
<td>mynn</td>
<td>มีนน์</td>
<td>มีนน(2)</td>
<td>ตะเภา(2)</td>
<td>minn(2)</td>
<td>minn(2)</td>
</tr>
<tr>
<td>harrod</td>
<td>แฮร/uni0E4C lowเริด</td>
<td>ฮาร/uni0E4C lowวูด</td>
<td>แชร/uni0E4C lowเริด</td>
<td>harold(2)</td>
<td>sherrod(2)</td>
</tr>
<tr>
<td>bowerman</td>
<td>โอบเออร/uni0E4C lowเมิ่น</td>
<td>โอบเออร/uni0E4C lowเมิ่น</td>
<td>โอบเออร/uni0E4C lowเมิ่น</td>
<td>overman(2)</td>
<td>overman(2)</td>
</tr>
<tr>
<td>blacka</td>
<td>แบคลา</td>
<td>แบล็คเคอร/uni0E4C low</td>
<td>บิลิก</td>
<td>blacker(23)</td>
<td>blacker(23)</td>
</tr>
<tr>
<td>pettus</td>
<td>เพตตัส</td>
<td>เพตตัส</td>
<td>foetus(2)</td>
<td>(1)</td>
<td>(1)</td>
</tr>
<tr>
<td>exmouth</td>
<td>เอกลัมมิก</td>
<td>สมูท</td>
<td>smoot(14)</td>
<td>(1)</td>
<td>(1)</td>
</tr>
<tr>
<td>hostess</td>
<td>โฮสเทส</td>
<td>โคสเทต</td>
<td>cost(3)</td>
<td>chesters(2)</td>
<td>(1)</td>
</tr>
<tr>
<td>site</td>
<td>ไซต</td>
<td>ซิต</td>
<td>city(4)</td>
<td>(1)</td>
<td>(1)</td>
</tr>
</tbody>
</table>

Table 5.3: Examples of where at least three models failed to identify correct transliterations at the first rank between English and Thai. The numbers in parentheses indicate the rank for the correct transliteration.
Table 5.4: Transliteration identification results for different combinations of context-dependent DBN models

<table>
<thead>
<tr>
<th>Combination</th>
<th>En-Ba</th>
<th>En-Ch</th>
<th>En-Hi</th>
<th>En-Ka</th>
<th>En-Ru</th>
<th>En-Ta</th>
<th>En-Th</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONs1+CONs2</td>
<td>0.967</td>
<td>0.791</td>
<td>0.864</td>
<td>0.856</td>
<td>0.980</td>
<td>0.859</td>
<td>0.811</td>
</tr>
<tr>
<td>CONs1+CONt1</td>
<td>0.971</td>
<td>0.800</td>
<td>0.865</td>
<td>0.861</td>
<td>0.987</td>
<td>0.864</td>
<td>0.834</td>
</tr>
<tr>
<td>CONs1+CONt2</td>
<td>0.971</td>
<td>0.864</td>
<td>0.868</td>
<td>0.866</td>
<td>0.984</td>
<td>0.872</td>
<td>0.865</td>
</tr>
<tr>
<td>CONs2+CONt1</td>
<td>0.969</td>
<td>0.846</td>
<td>0.865</td>
<td>0.863</td>
<td>0.984</td>
<td>0.971</td>
<td>0.856</td>
</tr>
<tr>
<td>CONs2+CONt2</td>
<td>0.970</td>
<td>0.883</td>
<td>0.870</td>
<td>0.867</td>
<td>0.984</td>
<td>0.874</td>
<td>0.879</td>
</tr>
<tr>
<td>CONt1+CONt2</td>
<td>0.965</td>
<td>0.811</td>
<td>0.863</td>
<td>0.854</td>
<td>0.984</td>
<td>0.865</td>
<td>0.845</td>
</tr>
<tr>
<td>ALLCONst</td>
<td><strong>0.972</strong></td>
<td>0.872</td>
<td>0.870</td>
<td><strong>0.868</strong></td>
<td>0.984</td>
<td><strong>0.877</strong></td>
<td>0.875</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Combination</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONs1+CONs2</td>
<td>0.979</td>
</tr>
<tr>
<td>CONs1+CONt1</td>
<td><strong>0.983</strong></td>
</tr>
<tr>
<td>CONs1+CONt2</td>
<td>0.982</td>
</tr>
<tr>
<td>CONs2+CONt1</td>
<td>0.980</td>
</tr>
<tr>
<td>CONs2+CONt2</td>
<td>0.980</td>
</tr>
<tr>
<td>CONt1+CONt2</td>
<td>0.979</td>
</tr>
<tr>
<td>ALLCONst</td>
<td><strong>0.983</strong></td>
</tr>
</tbody>
</table>

5.3 Experiments (NEWS 2009 and 2010 shared task data)

Combination of these four context dependent DBN models. The values in **bold** format in Table 5.4 represent the best overall result compared to the results in Table 5.2. The results in Table 5.4 seem to suggest that there is always a fair improvement (although slightly for some combinations) over the individual DBN models following the simple scheme of combining similarity estimates from the individual models. The results also suggest that the use of character context information from both the source and target strings will always result in better transliteration identification quality to the case when character context information from only the source or target string is used. This is also the case in (Kondrak and Sherif 2006) where an average context dependent DBN model and Pair HMMs outperform manually designed methods on the cognate identification task. In this section, we have only explored a small space of combinations. There are many DBN model combinations associated with the edit distance based DBN model framework that should be worth investigating. It should also be interesting to investigate the quality of transliteration identification quality from a combination of Pair HMM transliteration similarity estimates with those of the edit distance based DBN models.
5. Transduction-based DBN models for machine transliteration

5.4 Conclusion

In this chapter, we have applied a more generic DBN modeling approach in an experimental transliteration identification task. We successfully adapted three DBN templates associated with the approach to compute transliteration similarity. We specified several models from the DBN templates and evaluated them against the performance of the Pair HMMs on the same standard transliteration corpora for seven language pairs. Results from our transliteration identification setup show that of the three DBN templates, the template for context-dependent DBN models results in models that achieve the highest transliteration identification accuracy as compared to models associated with the other DBN templates. The context-dependent DBN models also achieve a considerable improvement in accuracy for all language pairs when compared to the standard baseline approach of pair n-gram models. This includes the case for English-Russian where the Pair HMMs did not perform well as we would have expected. The context-dependent DBN models also result in better transliteration identification accuracy compared to Pair HMMs for more than half of the language pairs. The results from the use of the DBN models in this chapter are generally similar to those that were reported for previous applications of the models in pronunciation classification (Filali and Bilmes 2005) and cognate identification (Kondrak and Sherif 2006). An analysis of the transliteration identification results associated with the use of the DBN models also suggests that using ensembles of DBN models to compute transliteration similarity may guarantee improvements in accuracy.

It is important to note that although we have investigated a reasonable number of edit distance based DBN models in our experimental transliteration identification task, the DBN modeling approach introduced in this chapter still offers a huge space of possible and feasible models which should be tested for use in transliteration identification.
Chapter 6

Applying DBN models in transliteration mining

6.1 Introduction

In the previous two chapters, we evaluated the application of the Pair Hidden Markov models (Pair HMMs) approach and the transduction-based Dynamic Bayesian Network (DBN) models in an experimental transliteration identification (TI) task. Specifically, the TI task datasets comprised a collection of source and target Named Entity (NE) pairs which we arbitrarily divided into reasonable training and test sets. We then used the trained models and determined from the transliteration similarity scores, the rank of the candidate target NE (for each source NE in the test set). In the TI experiments, we assumed that there were no ‘noisy’ entities in the datasets. We also assumed knowledge about NEs in one language (which for convenience we call a source language) for which we were supposed to determine a matching target NE from a list of candidate target NEs in the test set. We refer to this as an ideal transliteration identification setup.

When we use real-world data to identify transliterations, we expect both source and target sides of the data to have an additional set of ‘noisy entities’. This is the case in most transliteration mining tasks where real-world data is obtained from Web-based comparable corpora including news corpora (Klementiev and Roth 2006, Sproat et al. 2006, Khapra et al. 2010) and encyclopedic corpora (mainly Wikipedia) (Kumaran et al. 2010b). This Web-based data is created by a variety of users who represent it in a variety of ways leading to high proportions of noisy entities. In this chapter, we aim to further establish the value of the DBN models introduced in Chapters 4 and 5 by evaluating their application to mining transliteration pairs from

This chapter is an extended version of the following publications:
P. Nabende – Mining transliterations from Wikipedia using Dynamic Bayesian Networks, To appear in Recent advances in Natural Language Processing, Sept 2011, Hissar, Bulgaria.
real-world noisy data.

In our investigation of the application of DBN models in this chapter, we hereby distinguish between two transliteration mining related tasks in which the DBN models are evaluated. In the first task, we follow the NEWS 2010 transliteration mining shared task setup (Kumaran et al. 2010b) where bilingual Wikipedia topics extracted using Wikipedia inter-language links (WILs) are provided as raw data for mining single word transliteration pairs. During the shared task, we applied the Pair HMM with distinct transition and emission parameters (Figure 4.4) to mine English-Russian transliteration pairs. After the shared task, we extended the evaluation of more DBN models on the English-Russian dataset and on additional shared task datasets for two language pairs: English-Hindi and English-Tamil.

In the second transliteration mining task, we investigate the application of the DBN models in mining single word transliteration pairs from Wikipedia article content in addition to the paired topics. In the next section, we provide a further description of the two transliteration mining tasks. Later, we introduce the DBN model settings we have chosen to evaluate in the two related transliteration mining tasks.

6.2 Wikipedia – A source for transliteration mining

Wikipedia is an online encyclopedic resource in which over 270 languages have been used to write articles. In each language Wikipedia, there is a set of articles and for some articles access is provided to pages about the same topic in other languages. Figure 6.1 shows two Wikipedia articles about the same topic, one in English titled as “2010-2011 Middle East and North Africa protests” while the other is in Bangla (which is an eastern Indo-Aryan language native to the region of eastern South Asia known as Bengal) and is accessed using the Bangla Wikipedia inter-language link which exists on the English page as shown. Currently, there is a growing interest in using such article pairs as comparable text for extracting various types of information. In our case, we explore the use of such article pairs across writing systems for mining single word transliteration pairs. As mentioned in the Introduction section above, we divide the Wikipedia transliteration mining task into two sub tasks. The most common task (which for convenience, we refer to here as the ‘first’ Wikipedia transliteration mining task) uses Wikipedia’s link structure to mine bilingual NEs. Different types of links can be used for this task (Erdmann et al. 2008) including: inter-language links (an inter-language link is a link between two articles in different languages), redirect pages (page that has no content but only a link to a target page), and link texts (text part of the link that a user can click to reach a target page). In this chapter, we use only inter-language links following the setup for the NEWS 2010 shared task.
on transliteration mining (Kumaran et al. 2010b). In our ‘second Wikipedia transliteration mining task’, we use the main Wikipedia article content in addition to the paired topics. Although ‘linked text’ appears in main content, we do not distinguish it from the unlinked text. We therefore consider all named entities tagged in a preprocessing stage as candidate transliterations irrespective of whether they are part of linked text or not. As is depicted in Figure 6.1, it is clear that the main Wikipedia article content provides a lot more data that may result into the mining of many more transliteration pairs as compared to the case of using only Wikipedia’s article topics.

### 6.2.1 Transliteration mining using Wikipedia inter-language links

Recently, many studies have found it inexpensive to extract a lot of specific cross-language data including named entities and terminologies by using only Wikipedia’s inter-language links (Adafre and de Rijke 2006, Mohammadi and GhasemAghaei...
Applying DBN models in transliteration mining

Figure 6.2: Transliteration mining task and evaluation overview following the 2010 NEWS shared task. (Adapted from Kumaran et al., 2010b)

We will first use comparable Wikipedia topics that have been identified from the Inter-language links to constitute the collection of raw data to which the DBN models will be applied and evaluated for mining single word transliteration pairs. Figure 6.2 is an overview of the the NEWS 2010 transliteration mining shared task setup. We use the same setup used in the shared task to simplify our comparison of the application of the DBN models against state-of-the-art approaches that were evaluated (Kumaran et al. 2010a) on the same standard transliteration corpora.

In Figure 6.2, the ‘Interwiki links’ refer to the collection of article titles such as the English and Bangla titles in Figure 6.1 that are provided for the shared task as-is. Since manual extraction of transliteration pairs is used to create the Gold standard, only a small random sample of the many noisy article titles are provided as the gold set. Given the raw interwiki links, transliteration mining systems that participated in the shared task were supposed to return single word transliteration pairs from each pair of Wikipedia topics if they identified any, and otherwise, returned nothing if no transliteration pairs were identified. On the evaluation side of Figure 6.2, A refers to pairs from the transliteration mining system that are evaluated as True Positives.

6.3 DBN model selection for transliteration mining

(TP), B refers to those evaluated as False Positives (FP), C refers to those evaluated as False Negatives (FN), and D to those evaluated as True Negatives (TN). In section 6.4, we will refer to the notations in Figure 6.2 when describing the evaluation setup for this transliteration mining task.

6.2.2 Transliteration mining using comparable Wikipedia article text

In addition to using Wikipedia’s article titles to mine transliterations, we also explore the use of the main article content. The motivation here as suggested in Figure 6.1 is quite obvious, that we expect to find more named entities in the main article content than there are in the article titles. However, for the same number of Wikipedia topics, we expect the datasets to grow exponentially hence requiring more time and effort to pre-process and analyse. Instead of using all the article pairs between two languages for mining transliterations, we limited the size and identified only article pairs that we hypothesized to contain a sufficient number of transliteration pairs. To help us achieve that, we used as a starting point existing statistics about the number of page visits associated with the English Wikipedia for a given duration. Specifically, we used statistics about the English Wikipedia articles that got the highest number of page hits during the month of August 2009 and chose those that ranked highly in that regard. The corresponding articles in the other language were then easily retrieved through the inter-language links. Because there is need to manually annotate a gold standard from this type of data, for experimental purposes we limited the number of document pairs to enable the evaluation of the DBN models. Again, because of the requirement to manually prepare the gold standard dataset and the lack of expertise in most of the writing systems, we evaluated the DBN models for the English-Russian language pair only.

6.3 DBN model selection for transliteration mining

The DBN models that we apply to the transliteration mining tasks in this chapter have already been introduced in Chapters 4 and 5. In this section we present the specific DBN model structures we chose to apply in mining transliterations from Wikipedia. To explain the choice of model structures, we briefly review the results from the transliteration identification experiments in Chapters 4 and 5. Chapter 5 results show that the context-dependent DBN models outperform the other DBN models in most of the language pairs. It is therefore natural to consider them as the most suitable from the set of DBN models for application in mining transliterations from Wikipedia. However, Chapter 5 results also show that the best performing Pair
HMMs achieved accuracies and MRRs comparable to those for some of the context-
dependent transduction-based DBN models. The Pair HMMs even performed best in
the identification of transliterations between English and Hindi, and between English
and Kannada. Moreover, we also note that the application of the transduction-
based DBN models was limited by slow runtimes during inference for training the
models and for computing transliteration similarity. This drawback on the use of the
transduction-based DBN models could prove costly in a task involving the analysis
of huge amounts of data. This is indeed the case in the transliteration mining tasks
in this chapter where we apply the models to mine from hundreds of thousands of
Wikipedia paired topics.

The transliteration mining tasks in this chapter necessitate an evaluation of most
of the model settings for the Pair HMMs and transduction-based DBN models from
Chapters 4 and 5. However, the results from Chapters 4 and 5 suggest that the use
of some algorithms (for the case of Pair HMMs) and some model generalizations (for
the case of transduction-based DBN models) always lead to gains in transliteration
identification quality over the other Pair HMM algorithms and transduction-based
DBN model generalizations. Therefore, we have chosen to consider model structures
from both DBN approaches that achieved the highest transliteration identification
accuracy and MRR for any of the language pairs investigated in this chapter. In
the following two subsections, we specify the chosen DBN model structures for each
approach.

6.3.1 Pair HMMs

From the Pair HMM approach, we use the Pair HMM structures and the respective
algorithms that resulted in the best transliteration identification accuracy in
Chapters 4 and 5. The Forward log-odds algorithm for the Pair HMM with three
transition parameters (which we denote here as PHMM3 – see Figure 4.3) led to the
highest transliteration identification accuracy for most of the language pairs when
compared to other Pair HMM structures and algorithms. However, the other scoring
algorithms for PHMM3 resulted in the least transliteration identification accuracy
when compared to other Pair HMM variants where transition parameters were used
in computing transliteration similarity. We therefore decided to evaluate at least
two Pair HMM variants that had the highest transliteration identification accuracy
for a particular language pair. Based on that, we chose to use PHMM3 and the
Pair HMM variant that uses five transition parameters (denoted here as PHMM5 –
see Figure 4.2) with their respective forward log-odds algorithms for mining single
word transliteration pairs from English-Chinese and English-Tamil Wikipedia paired
topics. For English-Hindi, we use PHMM3 and the Pair HMM variant that uses
nine distinct transition parameters (denoted here as PHMM9 – see Figure 4.4). For English-Russian, we use PHMM5 and PHMM9.

### 6.3.2 Transduction-based context-dependent DBN models

From the transduction-based DBN model generalizations presented in Chapter 5, we also chose the best performing DBN models. On all language pairs context-dependent DBN models achieved the highest transliteration identification accuracy compared to other transduction-based DBN models. We have therefore chosen to use a context-dependent DBN model that models the dependency of the edit operation variable on the current and previous target string character (denoted here as CONt2) for mining transliterations from Wikipedia paired topics for three language pairs: English-Chinese, English-Hindi, and English-Tamil. For English-Russian we use a context-dependent DBN model that models a dependency of the edit operation on the current source character only (See section 5.2.3 for detailed description of the transduction-based DBN model generalizations). Table 6.1 is a summary of the selected Pair HMM and transduction-based DBN models.

<table>
<thead>
<tr>
<th>Language pair</th>
<th>PHMMm1</th>
<th>PHMMm2</th>
<th>FB-DBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-Chinese</td>
<td>PHMM3 forw. LO</td>
<td>PHMM5 forw. LO</td>
<td>CONt2</td>
</tr>
<tr>
<td>English-Hindi</td>
<td>PHMM3 forw. LO</td>
<td>PHMM9 forw. LO</td>
<td>CONt2</td>
</tr>
<tr>
<td>English-Russian</td>
<td>PHMM5 forw. LO</td>
<td>PHMM9 forw. LO</td>
<td>CONs1</td>
</tr>
<tr>
<td>English-Tamil</td>
<td>PHMM3 forw. LO</td>
<td>PHMM5 forw. LO</td>
<td>CONt2</td>
</tr>
</tbody>
</table>

Table 6.1: PHMMs and transduction-based DBN models for mining transliterations from Wikipedia. forw. LO refers to the forward log-odds algorithm for a given Pair HMM. PHMMm1 and PHMMm2 refer to the PHMMs that have the best and second best accuracy and MRR from Chapter 4 results.

### 6.4 Experiments using NEWS 2010 shared task setup

#### 6.4.1 Wikipedia inter-language link data

In this task, three sets of data were provided per language pair for the NEWS 2010 shared task on transliteration mining (Kumaran et al. 2010b). For each language pair, 1000 hand picked pairs of single word NEs were provided as seed data for training any proposed transliteration mining system. Then pairs of corresponding article topics that had been collected using Wikipedia Inter-language links (WILs)
from a given English Wikipedia data dump were provided as data from which single-word transliteration pairs were to be mined. For all language pairs, the Wikipedia paired topics were noisy with a large number of unwanted symbols which necessitated a data pre-processing step before applying any transliteration mining system. Most of the unwanted symbols included: temporal and numerical expressions, punctuation symbols, formatting symbols, mathematical operator symbols, and characters from other writing systems that are not part of the source or target language writing system. 1000 Wikipedia links were also randomly selected (from the large noisy WILs) from which single word NE pairs for each link were manually identified and annotated to constitute the gold standard set used to evaluate the application of the transliteration mining systems. The gold set also had cases where there existed no transliteration pairs.

For this particular task, we expect there to be no effect on the final result in using one language as the source language and the other language as the target and vice-versa since we do not have prior knowledge about either the source or target string. However, as in the transliteration identification tasks in Chapters 4 and 5, we use English on the target side while each of the other languages is used on the source side.

To reduce data sparseness, all datasets were converted to lowercase. For each of the DBN approaches, the source and target language alphabets used by the models were restricted to the characters that were found in the seed set for a first run of the models. Although the seed sets contained the standard characters for most of the languages, there were a lot of characters (mostly diacritics) in the noisy Wikipedia topics that were missing in the seed set. With the aim of determining how to evaluate words in the noisy set with the ‘new’ characters, we checked on the reference set and observed that the words in the gold standard data hardly used any characters apart from those in the seed set. We expect that an additional benefit of the transliteration mining task is to find new words including those that use uncommon characters which usually convey an unusual original pronunciation of the word. Thus, we also considered words with ‘new’ characters irrespective of how they were treated in the evaluations of the NEWS 2010 transliteration mining shared task (Kumaran et al. 2010b). Table 6.2 is a summary of the datasets indicating the total number of noisy Wikipedia paired topics for each language pair, and the final number of Wikipedia topic pairs that were used to build the set of candidate NEs per language pair.

Each dataset was transformed into a numerical representation to enable efficient computation using the training and string similarity estimation algorithms. For words that had ‘new’ in the Wikipedia topics, we used only information about the ‘known’ characters which were associated with the DBN model parameters and ignored the ‘unknown’ characters when computing transliteration similarity.
6.4 Experiments using NEWS 2010 shared task setup

<table>
<thead>
<tr>
<th>Language pair</th>
<th>before pre-processing</th>
<th>after pre-processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-Chinese</td>
<td>196047</td>
<td>175013 (89.3%)</td>
</tr>
<tr>
<td>English-Hindi</td>
<td>16963</td>
<td>14620 (86.2%)</td>
</tr>
<tr>
<td>English-Russian</td>
<td>345969</td>
<td>296053 (85.6%)</td>
</tr>
<tr>
<td>English-Tamil</td>
<td>13883</td>
<td>13249 (95.4%)</td>
</tr>
</tbody>
</table>

Table 6.2: Number of Wikipedia topic pairs before and after pre-processing

Acquiring additional training pairs - Iterative training

One critical issue in learning a transliteration model is the use of manually labeled training data. To ensure accurate learning, a large number of transliteration matches are needed. But for the shared task, only few are provided as seed data. One approach we use to automatically acquire additional unverified training pairs from the noisy Wikipedia data is to bootstrap from a seed set. This is described in the following paragraph.

Assuming that the noisy Wikipedia data constitutes our set of unlabeled data U. We first utilize the seed set to learn a transliteration model which for convenience we denote here by \( T_1 \). We apply \( T_1 \) to U to identify the set of transliteration pairs \( L_1 \) whose similarity score according to the transliteration model are above a selected threshold score \( th_1 \). We then add \( L_1 \) to S and use \( (S + L_1) \) to learn a new transliteration model which is applied to U again to identify an additional set of transliteration pairs \( L_2 \) whose similarity score according to the transliteration model is above a selected threshold value \( th_2 \). Following this approach we evaluate the performance of the resulting models for a given number of iterations. At each iteration, we test the learned transliteration model on the same gold standard set G.

6.4.2 Evaluation setup and results

The Pair HMM approach and the transduction-based DBN modeling approach are both generation-based approaches\(^1\) whereby, for each candidate transliteration pair, transliteration similarity is computed based on an underlying process which is perceived to generate the two words. Then, based on some criterion (which in our case is a threshold value), a decision is made as to whether the candidate transliteration pair qualifies as a transliteration pair.

\(^1\)The other common category of approaches that were evaluated in the NEWS 2010 shared task on transliteration mining (Kumaran et al. 2010b) are discriminative approaches which treat the mining task as a binary classification problem. The discriminative approaches necessitate the construction and use of a classifier to decide whether a candidate pair of words qualifies as a transliteration pair.
should be regarded by the transliteration system as a suggested true transliteration pair or not. In the following, we use a sample from the English-Russian Wikipedia topic pairs to demonstrate the evaluation setup used for the 2010 shared task on transliteration mining.

Table 6.3 shows the sample of English and Russian Wikipedia topics from which single word transliteration pairs are to be mined. Each underlined word in the Table has a transliteration match in the other language. Given the Wikipedia topics in Table 6.3, we expect the mining system to find transliterations pairs as illustrated in Table 6.4. Based on the sample results in Table 6.4, there are some important points to note with respect to the task. First, we see that not all Wikipedia article topics in Table 6.3 will result in a transliteration pair. This is illustrated in Table 6.4 where we use <NULL> to indicate that no transliteration pairs were found in some topic pair. Secondly, following the NEWS 2010 shared task setup (Kumaran et al. 2010a), morphological variations on either side are not regarded as transliterations. Thirdly, following the common definition for transliteration, translations having distinct pho-
netic transcriptions (a feature needed to qualify them as transliteration pairs) on either side are also not considered as transliterations. For example, although ‘District’ and ‘район’ in the first Wikipedia topic pair in Table 6.3 can be taken as translations of each other since they refer to a similar entity (pertaining to territorial administrative divisions) in the respective languages, their respective phonetic transcriptions /dɪstrɪkt/ and /raɪon/ (using the International Phonetic Alphabet (IPA)) are very different implying that they can not be treated as transliteration pairs. Finally, before training the models, we first convert the datasets (i.e. in the case of English and Russian) that have uppercase characters to lowercase.

a) Evaluation metrics

We train each Pair HMM using the Baum-Welch algorithm (Chapter 4), and each context-dependent DBN model using the generalized expectation maximization algorithm (Chapter 5). We then use the trained models to compute transliteration similarity between candidate transliteration pairs as described above. After applying each model, we inspect a subset of the transliteration similarity scores which lead us to the specification of different threshold values. If a low threshold value is used we get many suggestions of potential transliteration pairs returned by the mining system, and if a high threshold is used, we get very few suggestions. The collection of transliteration pairs whose transliteration similarity scores beat a given threshold value for each language pair are then evaluated against the gold set which contains a random selection of manually annotated topic pairs from the initial set of topic pairs.

The application of each approach in mining single word transliteration pairs from the Wikipedia paired topics is evaluated using three related metrics as described in Kumaran et al. (2010b, 2010a). The evaluation metrics are: Precision, Recall, and F-score. With reference to Figure 6.2, we compute the measures as follows:

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (6.1)
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (6.2)
\]

\[
\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6.3)
\]

\footnote{A true Russian transliteration for the English word ‘District’ is ‘Дистрикт’ and it is used to mean the same entity as in English. Conversely, a true English transliteration for the Russian word ‘район’ is ‘raion’ and it is also used to mean the same entity.}

\footnote{We use the $F_1$ score in our evaluations with respect to use for evaluation in the 2010 NEWS shared task on transliteration mining. But since, the aim of a transliteration mining task is to acquire only correct transliterations, it is more appropriate to adapt this measure so that more importance is attached to Precision than Recall.}
where:

- **TP (True Positives)** are word pairs that are identified as correct transliterations by the transliteration mining system and indeed are in the gold standard.

- **FP (False Positives)** are word pairs that are identified as correct transliterations but are incorrect transliterations according to the gold standard.

- **FN (False Negatives)** are word pairs that are identified as incorrect transliterations but are actually correct transliterations according to the gold standard.

- **TN (True Negatives)** are word pairs that are identified as incorrect transliterations and are indeed incorrect transliterations as per the gold standard. As seen in the expressions above we do not use these in the computations for Precision and Recall even when they are part of the result.

The different threshold values as introduced above enable us to plot Precision-Recall curves for a more general analysis of the transliteration mining results. However, when comparing the methods on single Precision, Recall, and F-score values, we set a threshold value that we think will result in an optimal discrimination between true transliteration pairs and non-transliteration pairs. In this case, We analyse a subset of the results returned by the system and then identify true transliteration pairs whose similarity scores we use to subjectively set the threshold value.

### b) English-Russian transliteration mining results

Table 6.5 shows the first 10 results out of a total of 15 for the transliteration mining methods that were evaluated on the English-Russian dataset. During the shared task we applied a Pair HMM with nine distinct transition parameters and distinct emission parameters (Figure 4.4). We use PHMM9_F to denote this Pair HMM in Table 6.5. When applying PHMM9_F, we used the forward algorithm to compute transliteration similarity. In Table 6.5, we have also included the result for a context-dependent DBN model which models a dependency of the edit operation variable on the current target character (CONt1). Since we did not apply the context-dependent DBN model during the shared task, its result is initially not included in the shared task report on the English-Russian dataset. However, we applied it to the same English-Russian datasets as used during the shared task. Table 6.5 specifies two types of runs: a **standard run** where only the seed datasets are supposed to be used to train the transliteration mining systems, and a **non-standard run** where participants are allowed to use additional and / or external data to complement the seed datasets. The results in Table 6.5 alone illustrate the varied applicability of the HMM framework. These results also suggest a comparable performance of the HMM-based methods to
### 6.4 Experiments using NEWS 2010 shared task setup

<table>
<thead>
<tr>
<th>Run type</th>
<th>Description</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>NED+</td>
<td>0.880</td>
<td>0.869</td>
<td>0.875</td>
</tr>
<tr>
<td>Standard</td>
<td>HMM + PC</td>
<td>0.813</td>
<td>0.839</td>
<td>0.826</td>
</tr>
<tr>
<td><strong>Standard</strong></td>
<td>CONt1</td>
<td>0.835</td>
<td>0.815</td>
<td>0.825</td>
</tr>
<tr>
<td>Non-standard</td>
<td>LFS + Seed+</td>
<td>0.797</td>
<td>0.853</td>
<td>0.824</td>
</tr>
<tr>
<td>*Standard</td>
<td>PHMM9_F</td>
<td>0.780</td>
<td>0.834</td>
<td>0.806</td>
</tr>
<tr>
<td>Standard</td>
<td>StringKernel</td>
<td>0.746</td>
<td>0.889</td>
<td>0.811</td>
</tr>
<tr>
<td>Standard</td>
<td>HMM</td>
<td>0.868</td>
<td>0.748</td>
<td>0.804</td>
</tr>
<tr>
<td>Standard</td>
<td>HMM + PC + IterT</td>
<td>0.843</td>
<td>0.747</td>
<td>0.792</td>
</tr>
<tr>
<td>Non-standard</td>
<td></td>
<td>0.730</td>
<td>0.870</td>
<td>0.790</td>
</tr>
<tr>
<td>Standard</td>
<td>DirecTL+</td>
<td>0.778</td>
<td>0.795</td>
<td>0.786</td>
</tr>
</tbody>
</table>

Table 6.5: Results of the NEWS 2010 transliteration mining shared task on English and Russian data (Source: Kumaran et al. (2010b)). During the NEWS 2010 shared task on transliteration mining, we applied PHMM9\_F which is the Pair HMM variant with distinct emission and transition parameters. Here, we used the forward algorithm to compute transliteration similarity. The two asterisks (**) indicate that the result for CONt1 (a context-dependent DBN model) was not included in the shared task report since it was evaluated at a later time after the shared task.

other state-of-the-art methods. The best method, NED+ (which is an extended form of the normalized edit-distance (NED) measure) is reported by Jiampojamarn et al. (2010) to be their baseline method. NED is defined as the ratio of the edit distance between two strings and the maximum length of the two strings. The relatively good performance of the NED+ measure and the HMM-based methods suggests that there may be no gain in using more complex methods to model transliteration similarity between English and Russian. The use of a context-dependent DBN model led to an improved F-score value over the Pair HMM approach but still below that for the simple NED+ approach.

c) English-Hindi transliteration mining results

For English-Hindi, we applied four Pair HMMs and one context-dependent DBN model. Figure 6.3 shows the Precision-Recall curves for the different models. In Figure 6.3, PHMM9\_F denotes the Pair HMM with distinct transition and emission parameters (Figure 4.4) and where the Pair HMM forward algorithm is used to compute transliteration similarity. PHMM9\_FLO refers to the same Pair HMM denoted by PHMM9\_F but where we use the Pair HMM forward log-odds algorithm to compute transliteration similarity. PHMM3\_FLO refers to the Pair HMM with
Applying DBN models in transliteration mining

Figure 6.3: Precision-Recall curves for Pair HMMs and a context-dependent DBN (CONDBN) model which models the dependency of the edit operation variable on the current and previous source characters after evaluation on 982 English-Hindi test items at different threshold values. PHMM9_F refers to the Pair HMM with distinct transition parameters (PHMM9) where we use the forward algorithm to compute transliteration similarity, PHMM9_FLO refers to PHMM9 where we use the forward log-odds algorithm. PHMM3_FLO refers to the Pair HMM with three transition parameters where we use the forward log algorithm, and PHMM3_IterT_FLO refers to the PHMM3_FLO where we train the model iteratively.

Three transition parameters between the edit states and where we use the Pair HMM forward log-odds algorithm to compute transliteration similarity. PHMM3_IterT_FLO refers to the Pair HMM with three transition parameters which we train iteratively as described in section 6.4.1 above for “acquiring additional training data”, and where we use the forward log-odds algorithm to compute transliteration similarity.

Figure 6.3 shows that the Pair HMMs where the forward log-odds algorithm is used to compute transliteration similarity (that is PHMM3_FLO, PHMM9_FLO,
and PHMM3_IterT_FLO) achieve a superior performance as their curves are closer to the upper right corner of the graph (where Precision and Recall is maximized) compared to the curves for PHMM9_F and the context-dependent DBN model.

This result is similar to the transliteration identification result for the English-Hindi language pair in Chapter 5 where a Pair HMM using the forward log-odds algorithm outperformed the context-dependent DBN models. The transliteration mining result in this case suggests that the approach of Pair HMMs could be suitable for mining transliteration pairs between English and Hindi.

In Table 6.6, we present our results for the Pair HMMs and context-dependent DBN model along with the first five shared task results for the approaches that were evaluated in mining English-Hindi transliteration pairs. The notations for the Pair HMMs and context-dependent DBN models are the same as defined above.

<table>
<thead>
<tr>
<th>Run type</th>
<th>Description</th>
<th>P</th>
<th>R</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Standard</td>
<td>PHMM3_FLO</td>
<td>0.930</td>
<td>0.976</td>
<td>0.952</td>
</tr>
<tr>
<td>* Standard</td>
<td>PHMM3_IterT_FLO</td>
<td>0.959</td>
<td>0.949</td>
<td><strong>0.954</strong></td>
</tr>
<tr>
<td>* Standard</td>
<td>PHMM9_FLO</td>
<td>0.934</td>
<td>0.975</td>
<td>0.954</td>
</tr>
<tr>
<td>* Standard</td>
<td>PHMM9_IterT_FLO</td>
<td>0.936</td>
<td>0.976</td>
<td><strong>0.955</strong></td>
</tr>
<tr>
<td>* Standard</td>
<td>CONs2</td>
<td>0.911</td>
<td>0.891</td>
<td>0.901</td>
</tr>
<tr>
<td>Standard</td>
<td>StringKernel</td>
<td>0.954</td>
<td>0.895</td>
<td>0.924</td>
</tr>
<tr>
<td>Standard</td>
<td>NED+</td>
<td>0.875</td>
<td>0.941</td>
<td>0.907</td>
</tr>
<tr>
<td>Standard</td>
<td>DirecTL+</td>
<td>0.945</td>
<td>0.866</td>
<td>0.904</td>
</tr>
<tr>
<td>Standard</td>
<td>(HMM+PC+IterT)+PC</td>
<td>0.953</td>
<td>0.855</td>
<td>0.902</td>
</tr>
<tr>
<td>Standard</td>
<td>BK-2007</td>
<td>0.883</td>
<td>0.880</td>
<td>0.882</td>
</tr>
</tbody>
</table>

Table 6.6: English-Hindi transliteration mining results for Pair HMMs and a context-dependent DBN model against NEWS 2010 Shared task results (Kumaran et al. 2010b). P refers to Precision while R refers to Recall. We again use an asterisk to indicate that we evaluated the respective models later after the shared task but on the same dataset.

As Table 6.6 shows, the Pair HMMs where we use the forward log-odds algorithm to compute transliteration similarity result in a considerably better F-score value compared to that for the best performing approach in the shared task. This result is quite surprising considering that we use the Pair HMMs in their most basic form. Although iterative training of the models (that is for PHMM3_IterT_FLO and PHMM9_IterT_FLO) leads to the best F-score values, there is only a slight

---

Note that the Pair HMM and CON DBN model results in Table 6.6 were not part of the NEWS 2010 transliteration mining shared task evaluation. Although we used the same datasets as those provided for the shared task, we conducted our English-Hindi and English-Tamil transliteration mining experiments at a later time.
increase over the case where we do iteratively train the same models (that is for PHMM3_FLO and PHMM9_FLO). Probably, the already high F-score values will be hard to improve using the same model structures with the same algorithms.

d) English-Tamil transliteration mining results

For English-Tamil, we present the results for only three models: two Pair HMMs and one context-dependent DBN model. The Pair HMMs, PHMM3_FLO and PHMM5_FLO are as defined in the previous subsection. Figure 6.4 shows the Precision-Recall curves for the three models. In Figure 6.4, we again see that the Pair HMMs achieve a superior performance as their curves are closer to the upper right corner of the graph compared to the curve for the context-dependent DBN model. The difference between the curves for the Pair HMMs and the context-dependent DBN model in Figure 6.4 is even bigger compared to the difference in Figure 6.3 for the English-Hindi results.

Figure 6.4: Precision-Recall curves for Pair HMMs and a context-dependent (CON) DBN model after evaluation on 690 English-Tamil test items at different threshold values.
6.5 Experiments using comparable Wikipedia article content

### Table 6.7: English-Tamil transliteration mining results for Pair HMMs and a context-dependent DBN model against NEWS 2010 shared task results (Kumaran et al. 2010b).

<table>
<thead>
<tr>
<th>Run type</th>
<th>Description</th>
<th>P</th>
<th>R</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Standard</td>
<td>PHMM3_FLO</td>
<td>0.913</td>
<td>0.966</td>
<td>0.936</td>
</tr>
<tr>
<td>* Standard</td>
<td>PHMM5_FLO</td>
<td>0.923</td>
<td>0.955</td>
<td>0.939</td>
</tr>
<tr>
<td>* Standard</td>
<td>CONs2</td>
<td>0.790</td>
<td>0.852</td>
<td>0.820</td>
</tr>
<tr>
<td>Standard</td>
<td>StringKernel</td>
<td>0.923</td>
<td>0.906</td>
<td>0.914</td>
</tr>
<tr>
<td>Non-Standard</td>
<td>LFS + Seed+</td>
<td>0.910</td>
<td>0.897</td>
<td>0.904</td>
</tr>
<tr>
<td>Standard</td>
<td>LFS</td>
<td>0.899</td>
<td>0.814</td>
<td>0.855</td>
</tr>
<tr>
<td>Standard</td>
<td>LFS</td>
<td>0.913</td>
<td>0.790</td>
<td>0.847</td>
</tr>
<tr>
<td>Standard</td>
<td>BK-2007</td>
<td>0.808</td>
<td>0.852</td>
<td>0.829</td>
</tr>
</tbody>
</table>

In Table 6.7, we present our results for the two Pair HMMs and the context-dependent DBN model alongside the first five shared task results for the approaches that were evaluated in mining English-Tamil transliteration pairs. As Table 6.7 shows, the Pair HMMs again result in better F-score values compared to the best performing approach in the shared task.

6.5 Experiments using comparable Wikipedia article content

6.5.1 Extracting training data from Wikipedia

For this set of experiments, we use Wikipedia’s topic pairs to automatically acquire a training set. Instead of searching for every possible topic pair, we restricted the scope of search to named entities, specifically person names, where we expect to find correct transliteration matches. We extracted name pairs by exploiting the structured nature of information in Wikipedia info-boxes (Bouma et al. 2009). We searched for entries that match a given pattern with respect to a language or country in Wikipedia categories such as “citizenship”, “nationality”, and “place of birth”, and then extracted the corresponding titles in the other language using inter-language links which are on the same Wikipedia page. Despite this restricted search, there were differences in the representation of names between the two languages and hence necessitating data pre-processing steps. First of all, we observed that while most English names started with the ‘first name’ (as in Barack Obama), many corresponding names in Russian started with a second name followed by a comma and lastly the first name (e.g. Обама, Барак). For these, we simply changed the order by swapping the strings preceding and following a comma for all the Russian names. Secondly, while only
the first and second name were used for one language, some named entity topics in the other language had a middle name/s/. To fix this problem, we first removed any abbreviations for middle names and considered only the first and last names. If the number of names was different between the source and target language topic pair, only the last name was taken if in the other language, we had one name in the topic pair. Lastly, some names had a hyphen in one (for example Shin-ichiro and Синъитиро) or both topics (for example Cary-Hiroyuki and Кэри-Хироюки). Names that had a hyphen in both topics were used as provided, but names that had a hyphen in only one topic (for example Chung-hee and Хи) were filtered out.

6.5.2 Comparable Wikipedia article content data

There are various criteria we can use to identify cross language Wikipedia article content for mining transliterations. However, it is important to select article pairs where we would expect to mine a reasonable number of transliteration pairs. We can consider, as a plausible premise, that a Wikipedia article with a large number of words will have a large number of named entities. We can then, evaluate its corresponding article in the other language to decide whether it also has a sufficient number of words from which we expect to get a large number of candidate named entities. In this case, we consider every entity in the content regardless of how the content is structured.

Given the two comparable Wikipedia articles, an estimation of the content size in them can lead us to a decision of using them for mining transliteration pairs or not. This approach to extracting comparable Wikipedia articles requires a search through the whole database of articles to get a ranking on article pairs with the highest number of words.

A different approach that we use in selecting Wikipedia article pairs for mining transliteration pairs, is to follow clues from Wikipedia’s statistics about articles. In our case, we use statistics associated with the most accessed English Wikipedia pages in a given month. We expect that a page that is visited very often, not only generates a lot of interest but also has a big content size. We also conjecture that the ‘interest’ associated with an article leads to corresponding representations about the same topic in other language Wikipedias. We also make a rough conclusion that its size in the other language should be reasonably large as well. Using statistics about the number of page hits per month for the English Wikipedia articles during the year 2009, we identified some 10 articles that had the highest number of page hits per day\(^5\) for the month of august 2009. We retrieved the corresponding Russian articles through the Inter-language links on the same page.

Given the English and Russian Wikipedia articles, we identified a variety of writ-

ten entities that were not relevant for transliteration mining. The irrelevant entities here include: temporal and numerical expressions, entities using characters from other writing systems that are different to the writing system for source or target language, punctuation symbols, and different formal expressions such as mathematical expressions. In the following we present the steps taken to filter out the irrelevant entities. First, we extracted only those words that were written using the Latin alphabet for English and the Cyrillic alphabet for Russian. A simple regular expression for this purpose is sufficient for removing most of the irrelevant entities. Secondly, we note that it may not be useful to consider every possible word from each of the articles as a candidate named entity. Instead, it could be more helpful if we identified types of words in the article pairs for which transliteration is commonly used. Named entities are well known to constitute the highest amount of unknown or out of vocabulary (OOV) words in a given application of a language, and it is often the case that transliteration is used to deal with them. Therefore, the transliteration mining process can be simplified, if we can identify and analyse only named entities. For the experiments in this section, we specified a regular expression for extracting words from each English Wikipedia article that started with an uppercase Latin character and was followed by lowercase character(s) irrespective of whether the word was linked or unlinked. The English named entities extracted in this way formed the set of candidate transliterations on the English side. It is important to note that depending on the named entity recognition (NER) requirements, the process of finding named entities from documents may not be as simple as specified here for our experiments, it may require the use of a sophisticated NER approach. For the Russian articles, we extracted only words that were written using Cyrillic characters (both lowercase and uppercase), also regardless of whether they were linked or unlinked text. From the set of English and Russian candidate named entities, we hand picked a subset of single word matches in the two languages to form the Gold standard set. Table 6.8 shows the article titles with the different sizes for the noisy data and gold standard data. As table 6.8 shows, the total number of English words is in most cases less than the total number of Russian words since we also assumed that words starting with lower case characters were to be treated as candidate named entities on the Russian side.

6.5.3 Evaluation setup and results

This task is similar to the first task the only difference that, here we analyse a higher number of candidate named entities from the article content. We will therefore use the same evaluation measures of Precision, Recall, and F-score as expressed in Equations 6.1, 6.2, and 6.3 respectively for the shared task evaluations. One plausible
6. Applying DBN models in transliteration mining

<table>
<thead>
<tr>
<th>English Wikipedia title</th>
<th>Total # words</th>
<th>gold standard size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>English</td>
<td>Russian</td>
</tr>
<tr>
<td>The Beatles</td>
<td>822</td>
<td>1693</td>
</tr>
<tr>
<td>Ted Kennedy</td>
<td>1360</td>
<td>517</td>
</tr>
<tr>
<td>Michael Jackson</td>
<td>1505</td>
<td>2277</td>
</tr>
<tr>
<td>YouTube</td>
<td>694</td>
<td>1026</td>
</tr>
<tr>
<td>Perseids</td>
<td>55</td>
<td>330</td>
</tr>
<tr>
<td>District 9</td>
<td>575</td>
<td>1857</td>
</tr>
<tr>
<td>Hans Christian Orsted</td>
<td>161</td>
<td>969</td>
</tr>
<tr>
<td>Inglorious Basterds</td>
<td>655</td>
<td>2271</td>
</tr>
<tr>
<td>Lady Gaga</td>
<td>852</td>
<td>1160</td>
</tr>
<tr>
<td>True Blood</td>
<td>880</td>
<td>732</td>
</tr>
<tr>
<td>Total of uniq words</td>
<td>4811</td>
<td>9334</td>
</tr>
</tbody>
</table>

Table 6.8: Wikipedia article data for English and corresponding Russian articles

evaluation approach is to compute the measures for different cut-offs on the number of high-ranking pairs of words that are returned after applying a given transliteration mining system. With respect to precision, using these cut-offs should give us a good impression of how well a particular approach does in ranking correct transliteration pairs before incorrect transliteration pairs (Manning and Schütze 1999). Table 6.9 shows the results at the different cut-offs after applying three Pair HMMs where the forward log-odds algorithm is used and a context-dependent DBN model which models a dependency of the edit operation variable on the current source character. As the table shows, the quality of the results from this task is much lower than that for the first task. This time we see that PHMM3 has much lower precision values compared with the other Pair HMMs. In this task where there is too much noise, it seems that the use of the log-odds ratio for computing transliteration similarity is sensitive to model changes. Table 6.9 also shows that the use of a context-dependent DBN model results in a stricter discrimination between true transliteration pairs and non-transliteration pairs for the first cut-off points where we expect the models to rank true transliterations higher than non-true transliterations. The result for the first two cut-off points reflects the result for the shared task setup where the use of a context-dependent model leads to an improved F-score value over the Pair HMM with distinct transition parameters (PHMM9). The results also suggest that for the DBN models to be valuable, we need to consider only a limited number of the high ranking transliteration pairs returned by the system. For example, as Table 6.9 suggests, we need to consider only the first 100 pairs for the setup in this section.
6.6 Conclusion

This chapter has showed the possibility of applying Pair HMMs and context-dependent DBN models in mining transliterations from real-world data (cross-language Wikipedia data in our case). The results from the first task associated with mining transliterations from Web-based cross-language topics, show an excellent performance by the Pair HMMs on the English-Hindi and English-Tamil datasets. The Pair HMMs also resulted in relatively better transliteration mining quality than the context-dependent DBN models on the two datasets. A comparison with methods that performed well in mining transliteration pairs from the inter-language topics in a recent transliteration mining shared task (Kumaran et al. 2010b) showed comparable and in some cases better F-score values. For the English-Russian dataset, the use of a context-dependent DBN model resulted in a higher F-score value than a Pair HMM using distinct transition parameters.

However, we see that the quality of transliteration mining results for the second task, where we use additional content mainly from the body of each of the cross language Wikipedia articles, is not as good as for the first task. Nonetheless, the results show that the Pair HMMs and context-dependent DBN model can still be

<table>
<thead>
<tr>
<th>Measure</th>
<th>Cut-off</th>
<th>PHMM3</th>
<th>PHMM5</th>
<th>PHMM9</th>
<th>CONs1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>100</td>
<td>0.650</td>
<td>0.860</td>
<td>0.750</td>
<td>0.920</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.530</td>
<td>0.755</td>
<td>0.630</td>
<td>0.760</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>0.423</td>
<td>0.663</td>
<td>0.497</td>
<td>0.603</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>0.340</td>
<td>0.555</td>
<td>0.383</td>
<td>0.500</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.294</td>
<td>0.468</td>
<td>0.308</td>
<td>0.416</td>
</tr>
<tr>
<td>Recall</td>
<td>100</td>
<td>0.246</td>
<td>0.326</td>
<td>0.284</td>
<td>0.348</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.402</td>
<td>0.572</td>
<td>0.477</td>
<td>0.576</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>0.481</td>
<td>0.754</td>
<td>0.564</td>
<td>0.686</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>0.515</td>
<td>0.841</td>
<td>0.580</td>
<td>0.758</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.534</td>
<td>0.886</td>
<td>0.583</td>
<td>0.788</td>
</tr>
<tr>
<td>F-score</td>
<td>100</td>
<td>0.357</td>
<td>0.473</td>
<td>0.412</td>
<td>0.505</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.457</td>
<td>0.651</td>
<td>0.543</td>
<td>0.655</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>0.450</td>
<td>0.706</td>
<td>0.528</td>
<td>0.642</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>0.410</td>
<td>0.669</td>
<td>0.461</td>
<td>0.602</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.380</td>
<td>0.613</td>
<td>0.403</td>
<td>0.545</td>
</tr>
</tbody>
</table>

Table 6.9: Precision, Recall, and F-score for different cutoffs of the number of returned English-Russian transliteration pairs

**6.6 Conclusion**
valuable if we consider up to an appropriate limit of the returned ranked list of potential transliteration pairs returned by the system.
Chapter 7

Applying Pair HMMs in transliteration generation

7.1 Introduction

This chapter extends our investigation of the application of Dynamic Bayesian Networks (DBNs) in transliteration generation as one of the other transliteration-related tasks. For the transliteration identification and mining tasks in the previous three chapters, we were required to identify transliteration pairs from a set of candidate named entities. For the transliteration generation task, we are required to convert an input NE from a ‘source’ language into one or more ‘target language’ NEs. Although results from the previous three chapters show that various Pair Hidden Markov Models (Pair HMMs) and transduction-based DBN models lead to high transliteration identification and mining accuracy, we need to determine whether Pair HMMs could be as valuable in a transliteration generation task. The transliteration generation task seems to be more challenging than the TI or TM tasks, since without knowledge about candidate NEs, the search space for a correct transliteration for a given input string is increased. The transliteration generation process relies on two main stages, each of which can easily affect output quality. The first stage is concerned with the segmentation of an input string and the second is concerned with the mapping of each segment to a correct representation in the target writing system. Although the latter stage can be addressed through a transliteration model such as the ones we are about to investigate, there is an increased level of ambiguity in mapping characters from one writing system to another as compared to the TI or TM tasks. In the transliteration generation process, the number of choices that we can associate with character relationships, can in theory be approximated to the product of the

This chapter is an extended version of the following publications:
size of the character vocabularies between the source and target language including transformations involving an empty symbol. It is also often the case that character transformations in writing systems could be one to many (for example, the Russian ⟨ч⟩ to the English ⟨ch⟩), and many to many. Recent work on transliteration generation does take this into account and alignment algorithms that induce different kinds of relationships have been developed (Jiampojamarn et al. 2010). In this chapter, our main interest is in the use of Pair HMM parameters for generating transliterations. Pair HMMs were initially designed for sequence alignment (Durbin et al. 1998), but their capacity to relate source and target elements using probabilistic values enables us to specify them as weighted finite state automata that we can use for transliteration generation. In that regard, we would like to know whether the use of Pair HMM parameters in the framework of finite state automata can lead to any gains in transliteration generation.

Traditionally, transliteration generation involves different writing systems, and we shall first test the models on standard transliteration generation corpora (Li et al. 2009, Li et al. 2010) where each language in a pair uses a different writing system. Apart from the traditional view of transliteration generation, we also introduce a task that according to our knowledge has not yet been addressed in the transliteration generation literature. We note that the output of cross-lingual applications when encountering ‘unknown’ words between two languages with the same writing system is affected in the same way as between two languages with different writing systems. We therefore propose the use of the traditional transliteration generation setup in dealing with ‘unknown’ words between languages that use the same writing system. In section 7.2, the two tasks are further described; in section 7.3.1, we introduce finite state automata concepts and show how to represent Pair HMMs as such; later we report on experiments with regard to the two tasks and a discussion of the respective results.

7.2 Transliteration generation tasks

7.2.1 Traditional machine transliteration task

The traditional transliteration generation task emphasizes the mapping of symbols from one writing system to symbols in another writing system. With reference to recent shared tasks on transliteration generation (Li et al. 2009, Li et al. 2010), each language pair involves languages that use different writing systems. In many cases, automatic transliteration generation involves a language that uses the Latin alphabet. A good case in point is again the recent shared tasks on transliteration generation where the English language is considered as either the source or target
7.2 Transliteration generation tasks

language per language pair. The vast amount of research on automatic transliteration generation for which the Latin alphabet is involved may be attributed to not only the Latin alphabet’s wide usage, but also to the simplicity that is associated with processing Latin characters. However, with recent advances on the part of the digital encoding of symbols from various writing systems (such as with the UTF8 encoding system), there is now some research on automatic transliteration generation between different writing systems that do not involve the Latin alphabet (Malik 2006, Malik et al. 2008). For this task, we shall experiment with standard transliteration corpora from the NEWS 2009 and NEWS 2010 shared tasks on transliteration generation (Li et al. 2009, Li et al. 2010).

7.2.2 Translating transliterations task

In this section, we argue that the transliteration generation task need not be restricted to the case where the source and target language use different writing systems. This is mainly because the limitations of NLP systems for which transliteration is used across writing systems are similar to those involving different languages that use the same writing system. Specifically, ‘unknown’ entities will affect a cross-lingual application that involves languages that use the same writing system in a similar way that they affect a cross-lingual application that involves different writing systems. However, to the best of our knowledge, the handling of ‘unknown’ entities in a cross-lingual application where the source and target language use the same writing system in the perspective of employing the traditional transliteration generation setup has yet to be addressed in literature. We specifically consider the situation where transliterated names originating from a different writing system differ across languages using the same writing system. Consider the examples in Table 7.1 where Russian names have been transliterated into: English, French, German, and Dutch. As can be seen, the transliterated names are spelt differently even though the four languages use the same alphabet. Such spelling variations for the same original name arise due to language specific differences.

<table>
<thead>
<tr>
<th>English</th>
<th>French</th>
<th>German</th>
<th>Dutch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexander Pushkin</td>
<td>Alexandre Pouchkine</td>
<td>Alexander Puschkin</td>
<td>Aleksandr Poesjkin</td>
</tr>
<tr>
<td>Nikita Khrushechev</td>
<td>Nikita Chruschtschow</td>
<td>Nikita Khrouchtchev</td>
<td>Nikita Chroesjtsjov</td>
</tr>
<tr>
<td>Yuri Andropov</td>
<td>Iouri Andropov</td>
<td>Juri Andropow</td>
<td>Joeri Andropov</td>
</tr>
<tr>
<td>Leonid Brezhnev</td>
<td>L’eonid Brejnev</td>
<td>Leonid Breschnew</td>
<td>Leonid Brezjnev</td>
</tr>
</tbody>
</table>

Table 7.1: Transliterated Russian names in four European languages that use the Latin alphabet.
The problems addressed in the traditional machine transliteration framework apply here as well. The different spelling variants try to match the underlying phonetic description for the original word and are very likely to be unknown by a cross-language application regardless of whether the languages involved use the same writing system or not. A dedicated module for transliterating the ‘unknown’ transliterated words is expected to help a cross-language processing system between languages using the same writing system in the same way a transliteration module is expected to improve performance across writing systems.

### 7.3 Using Pair HMMs in transliteration generation

Hidden Markov Models (HMMs) have been applied before in statistical machine translation to align words (Vogel et al. 1996). Just like the application of the classical HMMs to word alignment, we utilize the Pair HMMs for aligning characters and using the alignments to estimate parameters for transliteration generation. A transliteration generation module is obviously needed to help use the Pair HMM parameters as transliteration generation parameters. One natural option that we could start with is to develop our own module that can use Pair HMM parameters. The transliteration generation module would be required to facilitate the representation of the edit states associated with a Pair HMM and the corresponding emission parameters for transliteration generation. While Pair HMM emission parameters (Section 4.3) relate source and target string elements and could therefore be sufficient for modeling transliteration generation, the transliteration identification (TI) results of Chapter 4 suggested the importance of using Pair HMM transition parameters.\(^1\) Our transliteration generation module would also be required to incorporate Pair HMM transition parameters in order to test for their importance in a transliteration generation process.

Instead of developing our own transliteration generation module we follow a plausible approach of using Pair HMMs as finite state automata. We have already applied Pair HMMs in a manner similar to how a finite state automaton (acceptor) would be used to accept or reject input. Since a Pair HMM relates elements from source and target vocabularies, we can use them (Pair HMMs) for generation if they are represented as finite state transducers. As was introduced in Chapter 3, Pair HMMs find their origins as word alignment finite state automata, and the reverse process of representing them (Pair HMMs) as finite transducers should not be difficult to achieve. One advantage of following the approach of representing Pair HMMs as automata

\(^1\)The TI results showed a considerable improvement in transliteration identification quality with the use of Pair HMM transition parameters between edit states compared to the case when they are not used
7.3 Using Pair HMMs in transliteration generation

is that there already exist various software tools that enable the implementation of
different types of finite state automata and therefore our only task would be to spec-
ify how a given software tool would implement an automaton that corresponds to
a given Pair HMM. The requirements for transliteration generation that have been
noted in the previous paragraph for a transliteration generation module apply here
as well. In the next subsection we review the concepts associated with finite state
automata, and later we describe how we have represented Pair HMMs as finite state
transducers.

7.3.1 Finite state automata

A finite state automaton is defined as a mathematical abstraction which refers to a
model consisting of a finite number of states, a set of transitions between the states,
and actions (Jurafsky and Martin 2009). Each transition is described by a condition
that needs to be fulfilled so as to enable a state change, and an action describes an
activity that is to be performed at a given moment.

Finite state automata are broadly categorized into two: finite state acceptors
(FSAs) and finite state transducers (FSTs). The difference between the two is asso-
ciated with the representation on the transition arcs of the the automaton: an FSA
is used to only accept or reject input elements and an FST is used to map from an
input element to an output element. Hence, an FSA defines a model only for input
elements on the arcs and an FST defines a model that relates input and output el-
ements. Automata from each of these categories can be applied in a transliteration
generation framework. Of course, we can apply a finite state transducer to generate
a target string from some source string. And we can apply a finite state acceptor
to check whether the target string that was generated conforms to the spelling (or
pronunciation) regularities of the target language.

It is often convenient to represent finite state automata using state transition
diagrams where nodes denote states and edges are labeled with symbols. Figure 7.1
is a state transition diagram illustrating a finite state acceptor that will accept any
combination of the symbols ‘a’ and ‘b’ from a two symbol alphabet \{a,b\} if and only
if the first symbol is an ‘a’. In order to use a finite state acceptor, the following
parameters need to be defined (Jurafsky and Martin 2009):

- a finite set of states $Q = q_0, q_1, ..., q_n$
- a finite set corresponding to the input alphabet $\Sigma$
- the start state $q_0$
- the set of final states, $F \subseteq Q$
Applying Pair HMMs in transliteration generation

Figure 7.1: An example of a finite state acceptor that will accept only strings that start with the symbol ‘a’ using a two symbol alphabet \{a,b\}. Following usual convention, the start state \(q_0\) is represented with an incoming arrow and the final or accepting state is represented by the double circle.

- the transition function \(\delta(q, i)\) or transition matrix between states. The transition function defines a mapping to a new state \(q' \in Q\) given a state \(q \in Q\) and an input symbol \(i \in \Sigma\).

The transliteration generation task mainly requires the use of finite state transducers. A finite state transducer defines a relation between sets of strings and therefore enables a mapping from one representation to another. Jurafsky and Martin (2009) summarize four ways of perceiving an FST as follows:

- An FST as a recognizer. In this case an FST takes a pair of strings as input and outputs; and accepts the strings if the string-pair is in the string-pair language, and rejects if it is not. This view of a transducer can be adapted and applied to the transliteration identification and mining tasks in the previous three chapters.

- An FST as a generator. In this case the FST outputs pairs of strings of the language with a corresponding ‘yes’ or ‘no’.

- An FST as a set relator. In this case, an FST is used to compute relations between sets.

- An FST as a translator. In this case the FST reads a string and outputs another string. This view of an FST is the most suitable for fulfilling the transliteration generation requirements and it is the one we shall use for the task.

Of the four metaphors, the ‘FST as translator’ metaphor allows us to implement an FST as a transliterator. In the most basic case, a standard transliteration system can be represented as an FST. The incomplete Figure 7.2 illustrates a two state transducer where the start state is used to map characters from the Cyrillic alphabet...
7.3 Using Pair HMMs in transliteration generation

to the Latin alphabet using the post-2010 Passport system for romanizing Russian. In Figure 7.2, the arcs are labeled with input-output elements separated by a colon. In order to use a finite state transducer, the following parameters need to be defined (Jurafsky and Martin 2009):

- a finite set of states $Q = q_0, q_1, ..., q_n$
- a finite set corresponding to the input alphabet $\Sigma$
- a finite set corresponding to the output alphabet $\Delta$
- the start state $q_0 \in Q$
- the set of final states $F \subseteq Q$
- the transition function $\delta(q, w)$ between states. $\delta(q, w)$ returns a set of new states $Q' \in Q$ given the current state $q$ and an input string $w \in \Sigma^*$.
- the output function $\sigma(q, w)$ for determining the set of all possible output strings for each state. $\sigma(q, w)$ gives a set of output strings $o' \in \Delta^*$ given the current state $q \in Q$ and an input string $w \in \Sigma^*$.

Finite state automata of the type in Figure 7.1 and 7.2 can only be useful for a limited number of applications. In the transliteration generation process where we encounter a lot of ambiguity in mapping symbols from one writing system to another, there is need to use a probabilistic approach. A natural approach that is commonly used to achieve probabilistic modeling involves the augmentation of a finite state automaton such that each arc is associated with a probability representing the likelihood of taking a given path, and that the probability of all arcs leaving a given state sums to one. An FSA (respectively FST) that associates each arc with a probability is
referred to as a weighted finite state acceptor (WFSA) (respectively weighted finite state transducer (WFST)). WFSAs and WFSTs are formally defined as tuples over a semiring $\mathcal{K}$.

A WFSA is defined as a 7-tuple $\langle \Sigma, Q, q_0, F, E, \lambda, \rho \rangle$ where $\Sigma$, $Q$, $q_0$, and $F$ are as defined above. $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathcal{K} \times Q$, refers to the set of transitions; $\lambda \in \mathcal{K}$, refers to the initial weight; and $\rho : F \mapsto \mathcal{K}$, refers to the final weight function.

A WFST is a 8-tuple $\langle \Sigma, \Delta, Q, q_0, F, E, \lambda, \rho \rangle$ where $\Sigma$, $\Delta$, $Q$, $q_0$ and $F$ are as defined above. $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathcal{K} \times Q$, refers to the set of transitions; $\lambda \in \mathcal{K}$, refers to the initial weight; and $\rho : F \mapsto \mathcal{K}$, refers to the final weight function mapping final states to elements in $\mathcal{K}$.

In Chapter 2, we reviewed one of the earliest application of weighted finite state automata in transliteration generation, that is for Japanese Katakana to English back-transliteration (Knight and Graehl 1997). In some of the experiments we will test various weighted finite state automata while applying them in a manner similar to how they have been previously applied to generate transliterations. However, a major aim in this chapter is to determine the value associated with representing Pair HMMs as WFSTs and using the resulting Pair HMM-based WFSTs for transliteration generation.

### 7.3.2 Representing Pair HMMs as WFSTs

The emission states of a Pair HMM encode (like in a transducer) the relationship between source and target language elements. In order to use the Pair HMM parameters as transduction parameters, we first specify an FST structure which approximates that of a Pair HMM; and later specify the integration of Pair HMM emission and transition parameters on the arcs of the FST structure. Figure 7.3 shows a finite state automaton that approximates the Pair HMM with distinct emission and transition parameters (see Figure 4.4). In Figure 7.3, the Pair HMM emission parameters for a particular edit state are represented on the transition arcs that are directed towards a similar state in the automaton. The transition parameters between Pair HMM states are represented on corresponding transition arcs between similar states of the FST. Therefore, the probability that relates source and target elements in the FST of Figure 7.3 is the product of the Pair HMM transition probability from the previous state to the state (where the relationship is modeled) and the Pair HMM emission probability associated with the pair of symbols. For example, assuming that $p_{x_i,y_j}$ is an emission probability which relates the source element ($x_i$) and the target element ($y_j$) in the Pair HMM substitution state (M); the probability associated with relating these two symbols for a transition from the deletion state (X) to the match state (M) in the corresponding FST is specified as $(1 - \epsilon_X - \lambda_X - \tau_X) \times p_{x_i,y_j}$. In Figure 7.3,
7.3 Using Pair HMMs in transliteration generation

Figure 7.3: A finite state transducer approximation of the Pair HMM with nine transition parameters. The input and output elements separated by a colon are shown just before the parentheses on each arc of the transducer. $e$ is used to represent the empty symbol. The combination of Pair HMM transition and emission parameters are as shown on each arc. The emission probabilities ($p_..$) correspond to the emission probabilities in the Pair HMM edit states that model the relationship between the input and output elements.

we also define a start state to explicitly capture the starting parameters for the Pair HMM. In the TI and TM tasks, we assumed the Pair HMM to start in any of the three edit operation states using the transition parameters from the substitution state to each of the respective three edit states. We assume the same setup of starting parameters for the transliteration generation task.
7.4 Experiments using NEWS 2009-2010 shared task data

7.4.1 Data

The NEWS 2009-2011 shared tasks on transliteration generation datasets involved different writing systems between language pairs, with English being either a source or target language for each language pair. As shown in Table 7.2 we use seven of the 12 language pairs that were provided: English→Bengali, English→Hindi, English→Kannada, English→Russian, English→Tamil, English→Thai, and Thai→English.²

<table>
<thead>
<tr>
<th>Language pair</th>
<th>Training</th>
<th>Development</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>English→Bengali (en-ba)</td>
<td>13000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>English→Hindi (en-hi)</td>
<td>10014</td>
<td>2099</td>
<td>1000</td>
</tr>
<tr>
<td>English→Kannada (en-ka)</td>
<td>8065</td>
<td>2108</td>
<td>1000</td>
</tr>
<tr>
<td>English→Russian (en-ru)</td>
<td>5977</td>
<td>943</td>
<td>1000</td>
</tr>
<tr>
<td>English→Tamil (en-ta)</td>
<td>8037</td>
<td>2184</td>
<td>1000</td>
</tr>
<tr>
<td>English→Thai (en-th)</td>
<td>27668</td>
<td>1948</td>
<td>2000</td>
</tr>
<tr>
<td>Thai→English (th-en)</td>
<td>24051</td>
<td>1793</td>
<td>1994</td>
</tr>
</tbody>
</table>

Table 7.2: Size of training, development, and testing datasets per language pair. Source: NEWS 2009 and NEWS 2010 transliteration generation shared task data (Li et al. 2009, Li et al. 2010)

a) Data pre-processing

To help reduce on data sparseness, we ensured only lowercase representation for languages (English and Russian) where conversion to only lowercase was necessary. For some language pairs, some source language words had at least two transliteration variants in the training dataset. During training each variant was matched to the source word and used individually as a training pair. Some data sets also contained some multi-word sequences. We followed an approach similar to the one used by Finch and Sumita (2010) in handling multi-word sequences. For those multi-word sequences where the number of words in the source and target word sequences matched,

²The other language pairs from the two shared tasks include: English→Chinese, Chinese→English, English→Japanese Katakana, English→Japanese Kanji, and English→Korean Hangul. Apart from the English→Chinese and Chinese→English datasets, we did not use the other three datasets because they were expensive to purchase.
7.4 Experiments using NEWS 2009-2010 shared task data

we split the word sequences into individual words; during training, we matched words in the same word position in the source and target word sequence and used the resulting individual word pairs as training data; during testing, each source word from the multi-word sequence was transliterated individually, and the \(n\)-best transliteration lists associated with all the individual words in the source word sequence were subsequently combined into a single output transliteration sequence. For the other multi-word sequences in the training data where the numbers of words in the source and target word sequences differed, we introduced a \(\langle\text{space}\rangle\) token into the sequence, and treated it as one long sequence.\(^3\)

### 7.4.2 Transliteration models

a) WFST Parameter estimation

We apply two sets of WFSTs to transliteration generation. In the first set of WFSTs, we use apply WFSTs in the usual way. In the second set of WFSTs we use Pair HMM parameters to derive a WFST. However both sets of WFSTs use the notion of edit distance.

We varied the first set of WFSTs in terms of the states and their possible emissions. For one of these WFSTs, we use a structure corresponding to that of a Pair HMM (Figure 7.4). As Figure 7.4 shows, this WFST also uses separate states for substitutions, insertions and deletions. In this type of WFST, we introduce some kind of model bias by restricting the type of emissions to be of a certain kind at each state. In a different setting, we removed this bias by allowing all possible types of emissions (including insertions on the source and target side) from any state of the WFST. The idea in this case is to let the training procedure decide how to make use of the hidden layer of states without defining the function of each state. This is basically a test to see if the forward-backward parameter estimation procedure (which is used for training the WFSTs) is capable of learning some underlying structure which is not given to the system when training its parameters. However, we still have to define the number of states to be used in the WFST before training its parameters. In our experiments, we applied WFSTs with one to five states (excluding start and end state) and a fully connected graph with uniform initial settings. Furthermore, we also ran the training procedure with three additional randomly chosen initial parameters. For this set of WFST models, we used a publically available finite state automata toolkit called CARMEL\(^4\) for parameter estimation.

\(^3\)The other option that we could have followed for multi-word sequences where the number of words differed would have been to just ignore them during training as is the case in Jiampojamarn et al.’s work (2010).

\(^4\)http://www.isi.edu/licensed-sw/carmel
For the second set of WFST models, we first estimate Pair HMM parameters as described in Chapter 3; the Pair HMM parameters are then transformed into WFST transliteration generation parameters as described above and represented in a format suitable for use by the CARMEL software toolkit.

b) Phrase-based statistical machine translation

In addition to the WFSTs, we also tested the use of a phrase-based statistical machine translation approach on the English→Russian language pair. Phrase-based statistical machine translation (PSMT) is the current state of the art in data-driven machine translation, and has recently been applied to transliteration generation (Matthews 2007, Finch and Sumita 2008). It is based on the well-known IBM models
which are trained on large parallel corpora but use bilingual phrase tables instead of word link probabilities and fertility parameters. In the PSMT approach, various components are usually combined in a log-linear model (translation models, reverse translation model, word and phrase penalties, language models, distortion parameters, etc) with weights optimized using minimum error rate training (MERT). Various tools are available for training such a model and “decoding” (translating) input strings according to the model. In our case, we used the publically available toolkit Moses (Koehn et al. 2007) with its connected tools: GIZA++ (Och and Ney 2003) and IRSTLM (Federico et al. 2008). As a requirement of the transliteration generation task, we split the names on the character level. Machine translation applications usually require output word / phrase re-ordering. But for character-based translation / transliteration, we expect a monotonic ordering in the output that corresponds to the input. We therefore ensured that the PSMT system uses monotonic decoding. We left the other parameters for the Moses PSMT decoder unchanged. The model therefore uses the standard settings for character alignment with GIZA++, standard heuristics for the extraction and scoring of phrase alignments (character n-grams with a maximum length of 7 characters) and standard settings for the minimum error rate training (MERT) when tuning the models. The language model for the English-Russian case is a 5-gram model which we estimated from the target language side of the training dataset using the Witten-Bell smoothing technique which is implemented in the IRSTLM toolkit. There are various fixed parameters that can be tuned in the PSMT models. Among others, we could change the maximum size of phrases to be considered, various phrase extraction techniques can be used and language model parameters can be modified. In our setup, we did not tune these training specific parameters.

A major advantage of the PSMT approach over the weighted finite state transducers described above is that the extracted phrase tables (character n-grams) cover a lot of contextual dependencies found in the data. By exploiting these, we hope to find better transformations by translating sequences of characters instead of single characters. Furthermore, we do not have to model insertions and deletions explicitly but leave it to the translation table to change the lengths of translated strings. Another advantage is the explicit inclusion of a target language model to weight the possible outcomes of the system. In the transducer model, this is not easily possible as we include deletion operations. The reason being that the language model would always prefer shorter strings and therefore force the system to over-use the deletion operations when transforming strings. Of course, we do not expect the WFSTs to perform better than the PSMT approach in the transliteration generation tasks. The use of a PSMT system, however, gives us an idea of the extent to which transliteration generation quality is affected by the limitations of the WFSTs and whether
7. Applying Pair HMMs in transliteration generation

trying to address some of these limitations could lead to improved quality.

7.4.3 Evaluation metrics

We follow the same evaluation setup as used for the NEWS 2009 and NEWS 2010 shared tasks on transliteration generation. For each source language word in the test set, a participating system was required to generate and submit 10 best candidate transliterations. For cases where a source language word had alternative transliterations, all the alternatives were treated equally in the evaluation process. In the NEWS 2009 shared task on transliteration generation, six measures were used to evaluate transliteration generation quality. These include (Li et al. 2009): accuracy, fuzziness in Top 1 (mean F-score), mean reciprocal rank (MRR), mean average precision for reference transliterations (MAP_{Ref}), mean average precision in 10 best candidate transliterations (MAP_{10}), mean average precision for the system (MAP_{sys}). We use the same notation in Li et al. (2009) to define the evaluation metrics:

- \( N \): total number of names (source words) in the test set
- \( n_i \): number of reference transliterations for \( i^{th} \) name in the test set \( (n_i \geq 1) \)
- \( r_{i,j} \): \( j^{th} \) reference transliteration for \( i^{th} \) name in the test set \( (1 \leq j \leq n_i) \)
- \( K_i \): number of candidate transliterations produced by a transliteration system
- \( c_{i,k} \): \( k^{th} \) candidate transliteration (output by the transliteration system) for the \( i^{th} \) name in the test set \( (k \leq K_i) \).

**a) Word accuracy in Top-1 (ACC)**

Also known as the word error rate, it measures the correctness of the first transliteration candidate in the \( n \)-best candidate list produced by a transliteration system. ACC = 1 means that all top candidates are correct transliterations, that is, they match one of the references, and ACC = 0 means that none of the top candidates are correct.

\[
\text{ACC} = \frac{1}{N} \sum_{i=1}^{N} \{1 \text{ if } \exists r_{i,j} : r_{i,j} = c_{i,1}; \text{ 0 otherwise}\} \quad (7.1)
\]

**b) Fuzziness in Top-1 (mean F-score)**

The mean F-score measures how different, on average, the top transliteration is from its closest reference. F-score for each source word is a function of Precision and Recall and equals 1 when the top candidate matches one of the references and 0 when there
are no common characters between the candidate and any of the references. Precision and Recall are calculated based on the length of the longest common subsequence (LCS) between a candidate and a reference:

$$\text{LCS}(c, r) = \frac{1}{2} (\text{length}(c) + \text{length}(r) - \text{ED}(c, r))$$  \hspace{2cm} (7.2)$$

where \( \text{ED}(c, r) \) is the edit distance. For example, the longest common subsequence between “abcd” and “afcde” is “acd” and its length is 3. The best matching reference, that is, the reference for which the edit distance has the minimum is used. If the best matching reference \( r_{i,m} \) is given as \( r_{i,m} = \text{argmin}_i \) (ED(c_i,1,r_i,j)), Recall (R_i), Precision (P_i), are calculated as follows:

$$R_i = \frac{\text{LCS}(c_i,1,r_{i,m})}{\text{length}(r_{i,m})} \hspace{2cm} P_i = \frac{\text{LCS}(c_i,1,r_{i,m})}{\text{length}(c_i,1)}$$

The F-score is computed as the harmonic mean of Precision and Recall (see Equation 6.3), sometimes referred to as \( F_1 \) score.

c) Mean reciprocal rank

Measures the traditional MRR for any correct transliteration produced by the system, from among \( n_i \) candidates. 1/MRR tells approximately the average rank of the correct transliteration. MRR closer to 1 implies that the correct answer is mostly produced close to the top of the \( n \) best lists. If a candidate that matches one of the references is in the \( j^{th} \) position in the \( n \)-best list, its rank equals \( j \) and its reciprocal rank equals \( 1/j \). When none of the candidates matches any of the references, the reciprocal rank of the “matching” candidate is 0.

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \left\{ \min_j \frac{1}{j} \text{ if } \exists r_{i,j}, c_{i,k} : r_{i,j} = c_{i,k}; 0 \text{ otherwise} \right\}$$  \hspace{2cm} (7.3)$$

d) Mean Average Precision_{reference} (MAP_{reference})

Measures tightly the precision in the \( n \)-best candidates for the \( i^{th} \) source name, for which \( n_i \) reference transliterations are available. If all the references are produced, then MAP is 1.

$$\text{MAP}_{ref} =$$

$$\frac{1}{N} \sum_{i=1}^{N} \frac{1}{\min(n_i,10)} \left( \sum_{k=1}^{\min(n_i,10)} \frac{\text{number of correct candidates for } i^{th} \text{ word in } k\text{-best}}{k} \right)$$
e) **Mean Average Precision_{10} (MAP_{10})**

MAP_{10} measures the precision in the 10 best candidates for the \(i^{th}\) source name provided by the candidate system. In general, the higher \(\text{MAP}_{10}\) is, the better is the quality of the transliteration system in capturing multiple references. The number of reference transliterations may be more or less than 10. If the number of reference transliterations is less than 10, then \(\text{MAP}_{10}\) can never be equal to 1. Only if the number of reference transliterations for every source word is at least 10, then \(\text{MAP}_{10}\) could possibly be equal to 1.

\[
\text{MAP}_{10} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{10} \left( \frac{1}{\sum_{k=1}^{10} \text{number of correct candidates for } i^{th} \text{ word in } k\text{-best}} \right)
\]

f) **Mean Average Precision_{system} (MAP_{sys})**

\(\text{MAP}_{\text{sys}}\) measures the precision of the top \(K_i\)-best candidates produced by the system for the \(i^{th}\) source name, for which \(n_i\) reference transliterations are available. This measure allows the systems to produce a variable number of transliterations, based on their confidence in identifying and producing correct transliterations. If all the \(n_i\) references are produced in the top-\(n_i\) candidates (that is \(K_i = n_i\), and all of them are correct), then \(\text{MAP}\) is 1.

\[
\text{MAP}_{\text{sys}} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{K_i} \left( \frac{1}{\sum_{k=1}^{K_i} \text{number of correct candidates for } i^{th} \text{ word in } k\text{-best}} \right)
\]

7.4.4 Results

a) **English→Bengali, English→Hindi and English→Kannada results**

For English→Bengali, English→Hindi and English→Kannada, we applied three Pair HMM-based WFST models as shown in Table 7.3. phmm0wfst refers to the WFST that captures only the emission parameters of a Pair HMM. We applied phmm0wfst with the aim of determining whether using only Pair HMM emission parameters would suffice for the transliteration generation task. phmm5wfst refers to the WFST model that captures the parameters of a Pair HMM that uses five transition parameters (See Figure 4.2). phmm9wfst refers to the WFST model that captures the parameters of a Pair HMM that uses distinct transition parameters (See Figure 4.4).

Table 7.3 shows the results for three language pairs associated with the use of Pair HMM parameters in **standard runs** for the respective datasets. A **standard run** in this case refers to the use of only the training data that was provided for the NEWS
7.4 Experiments using NEWS 2009-2010 shared task data

<table>
<thead>
<tr>
<th>Language pair</th>
<th>model</th>
<th>accuracy</th>
<th>F-score</th>
<th>MRR</th>
<th>MAP_{ref}</th>
</tr>
</thead>
<tbody>
<tr>
<td>English→Bengali</td>
<td>phmm0wfst</td>
<td>0.021</td>
<td>0.641</td>
<td>0.035</td>
<td>0.021</td>
</tr>
<tr>
<td></td>
<td>phmm5wfst</td>
<td>0.100</td>
<td>0.713</td>
<td>0.135</td>
<td>0.100</td>
</tr>
<tr>
<td></td>
<td>phmm9wfst</td>
<td>0.100</td>
<td>0.713</td>
<td>0.132</td>
<td>0.100</td>
</tr>
<tr>
<td>English→Hindi</td>
<td>phmm0wfst</td>
<td>0.009</td>
<td>0.614</td>
<td>0.017</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>phmm5wfst</td>
<td>0.030</td>
<td>0.654</td>
<td>0.052</td>
<td>0.030</td>
</tr>
<tr>
<td></td>
<td>phmm9wfst</td>
<td>0.030</td>
<td>0.654</td>
<td>0.050</td>
<td>0.030</td>
</tr>
<tr>
<td>English→Kannada</td>
<td>phmm0wfst</td>
<td>0.010</td>
<td>0.621</td>
<td>0.012</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>phmm5wfst</td>
<td>0.015</td>
<td>0.614</td>
<td>0.021</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td>phmm9wfst</td>
<td>0.015</td>
<td>0.614</td>
<td>0.030</td>
<td>0.015</td>
</tr>
</tbody>
</table>

Table 7.3: Standard run transliteration generation results for three language pairs. phmm0wfst is the WFST that uses parameters from a Pair HMM with no transition parameters between edit states; phmm5wfst is a WFST that uses parameters from a Pair HMM with five transition parameters (Figure 4.2); and phmm9wfst uses parameters from a Pair HMM with distinct emission and transition parameters (Figure 4.4).

2010 shared task on transliteration generation. There is also a non-standard run, where the participants can use additional external datasets. The results in Table 7.3 suggest that Pair HMM parameters based on one-to-one character alignments are not good at all for transliteration generation. None of the Pair HMM-based WFST models achieve accuracies comparable to those of the systems that participated in the NEWS 2010 shared task with respect to the three language pairs. A review of the participating systems in the NEWS 2010 shared task puts the 'basic' application of Pair HMM parameters at a clear disadvantage. All the participating systems including a phrase-based statistical machine translation (PSMT) approach (Finch and Sumita 2010), n-gram and joint source channel - based models (Das et al. 2010), and an online prediction sequence prediction model based on many-to-many alignments modeled far more information than that represented by the Pair HMMs. In one way or another, all of the models used by the participating systems captured contextual information to some extent which was not the case for Pair HMM - based WFSTs in Table 7.3. However, the LCS-based F-score values hold some promise for the use of Pair HMMs in transliteration generation. We can see that even for the worst performance on accuracy, none of the Pair HMM - based models had below 60% F-score. The table also shows that the use of Pair HMM transition parameters between edit states leads to an improvement in transliteration generation quality over the case when they are not used.
7. Applying Pair HMMs in transliteration generation

b) English→Russian transliteration generation results

For English→Russian transliteration generation, we also present results for models that participated in the NEWS 2009 shared task on transliteration generation (Li et al. 2009). Table 7.4 shows results from the application of different WFST models and the PSMT approach. The results for the models that participated are marked with an asterisk. The models with the extension _rules refer to the case where we modeled for English vowel bi-gram combinations and bi-grams associated with Cyrillic romanization, and post-processing step that involved the use of a few transformation rules. When using development data, a check on the transliterations that were generated using Pair HMM parameters when applied in the most ‘basic’ way showed consistent mistransliterations. For example, in all cases where the Russian character л ‘l’ precedes the Russian soft sign ъ ‘ъ’, the Russian soft sign was missing. For example крёфельд and билбао were generated instead of крёфельд ‘krefeld’ and билбао ‘bilbao’ respectively. This affected transliteration generation quality. For such cases, simple transformation rules such as “л→лъ” were used on the generated transliterations in a post processing step. 25 transformation rules were specified to help deal with some of the mistransliterations. The Moses_PSMT system was used

<table>
<thead>
<tr>
<th>Model</th>
<th>accuracy</th>
<th>F-score</th>
<th>MRR</th>
<th>MAP_{ref}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard runs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>phmm0wfst</td>
<td>0.055</td>
<td>0.758</td>
<td>0.069</td>
<td>0.055</td>
</tr>
<tr>
<td>phmm9wfst</td>
<td>0.298</td>
<td>0.856</td>
<td>0.346</td>
<td>0.298</td>
</tr>
<tr>
<td>phmm5wfst*</td>
<td>0.293</td>
<td>0.845</td>
<td>0.325</td>
<td>0.293</td>
</tr>
<tr>
<td>phmm5wfst_rules*</td>
<td>0.354</td>
<td>0.869</td>
<td>0.394</td>
<td>0.354</td>
</tr>
<tr>
<td>Moses_PSMT*</td>
<td>0.509</td>
<td>0.908</td>
<td>0.619</td>
<td>0.509</td>
</tr>
<tr>
<td>Non-standard runs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>phmm5wfst*</td>
<td>0.341</td>
<td>0.776</td>
<td>0.368</td>
<td>0.341</td>
</tr>
<tr>
<td>phmm5wfst_rules*</td>
<td>0.515</td>
<td>0.821</td>
<td>0.571</td>
<td>0.515</td>
</tr>
<tr>
<td>edit_WFST*</td>
<td>0.321</td>
<td>0.768</td>
<td>0.403</td>
<td>0.321</td>
</tr>
<tr>
<td>edit_WFST_rules*</td>
<td>0.466</td>
<td>0.808</td>
<td>0.525</td>
<td>0.466</td>
</tr>
<tr>
<td>Moses_PSMT*</td>
<td>0.612</td>
<td>0.845</td>
<td>0.660</td>
<td>0.612</td>
</tr>
</tbody>
</table>

Table 7.4: Results for English→Russian transliteration generation. Models marked with an asterisk participated in the NEWS 2009 shared task on transliteration generation. phmm0wfst, phmm5wfst, and phmm9wfst are as defined in the caption of the previous Table. edit_WFST refers to the WFST that uses separate states for the different edit operations as shown in Figure 7.4. the extension _rules indicates the additional use of transformation rules in a post-processing step.
7.5 Experiments on translating transliterations

In this set of experiments, we investigate the application of transliteration models similar to those that we applied in the previous section. Here we evaluate the models on translating transliterations between English and Dutch, and between English and French. For some models, we investigate further settings in addition to those that have been used in the previous subsection.

7.5.1 Data

The datasets for this set of experiments were extracted from an English Wikipedia data dump from 2008/07/24 in a similar way as was done for collecting training named entities (NEs) for the second transliteration mining task in Chapter 5. In this case, we also used simple patterns to identify Russian names looking at the structured information in Wikipedia info-boxes. We looked at entries that match the pattern (Russian|Russia|Soviet) in categories such as “citizenship”, “nationality” and “place of birth”. Translations of these names are taken from the Wikipedia inter-language links (WILs) which exist on every source page. We collected all names potentially from Russian origin and their correspondences in other languages. We saved all name pairs for the language pairs we were interested in, performing some extra normalization similar to that described in Chapter 5. This includes: normalizing of names that had abbreviations (e.g. “George H.W. Bush”) and / or a middle name (e.g. “Nikita Sergejewitsch Chruschtschow”); and switching the order of first and family names (e.g. “Clinton, William Jefferson” instead of “William Jefferson Clinton”). For the language pairs in this case, the data set which we extracted in this way was also small.

with the settings described in subsection c.) above on ‘Phrase-based statistical machine translation’. Although the performance of the Pair HMMs is lower than that for the PSMT system, the results in Table 7.4 show that the use of contextual information in the Pair HMMs and application of contextual rules in a post-processing stage improve transliteration generation quality. In both cases the F-score for Pair HMMs that use the additional information approaches that of the PSMT system. Again these results prove that Pair HMM transition parameters are important for transliteration generation. For the non-standard runs we used additional English-Russian data from the geonames database (Chapter 3) to train the models. As the results show, the use of additional data leads to a general improvement in transliteration generation quality for all models, and the Pair HMM - based method where some context is used with post-processing transformation rules results in a relatively greater improvement.
We obtained 199 pairs of names for English-Dutch and 372 pairs for English-French. We did not manually check them and, therefore, our database includes names which are not typically Russian (such as Marc Chagall, born in the Russian empire as a son of a Jewish family). However, we assume that there are only very few of these exceptions. From each of our datasets, we removed 50 name pairs to form our two test sets for both language pairs. Each of the two test sets is used for evaluating all the models that are tested in this task. The remaining pairs were used for training and / or tuning model parameters.

### 7.5.2 Transliteration models

#### a) WFST parameter estimation

The WFSTs in this set of experiments are trained just as described in the previous task. For this set of experiments, we ran the training procedure with a uniform initial model and five other randomly chosen initial models which are aimed at reducing the likelihood of ending up in a suboptimal model. We first applied the edit distance model (see Figure 7.3) which implements separate states for substitutions, insertions, and deletions. Then, we also applied various FSTs where we varied the number of states while letting the training procedure decide on how to utilize the hidden layer of states. We also modified the input and output alphabets by changing the way of splitting strings into symbol sequences. Previously, we simply used character sequences for training and testing. For this task, we split the words into sequences of vowel or non-vowel n-grams. The training procedure for the latter case is similar to that in the previous cases.

#### b) Phrased-based statistical machine translation

For this task, we concentrated on modifying the PSMT models in the following ways: firstly, we changed the training data in such a way that the set for tuning is part of the training set instead of keeping a separate set for tuning. In our basic setting, we remove 50 additional name pairs from the training set to be used for tuning the SMT model parameters. In another setting, we simply used them for training as well. Here, we were interested in seeing how increasing the training set influences the performance before training (especially with our tiny training set). Furthermore, we would also like to know if tuning on parts of the training set may still lead to improvements on the test set.

Secondly, we changed the pre-processing step from character splitting to vowel/non-vowel splitting as described in the previous subsection for the WFST models. Here, we do not expect a similar effect on the results as we expect for the WFSTs. This
is because contextual information is already integrated in the phrase-based SMT model to a large extent and important character combinations already appear in the extracted phrase table with appropriate scores.

A last modification we investigated is the application of a larger language model. It is well-known that SMT models produce better results in general when increasing the language model. However, the transliteration task is different from the sentence translation task for which the Europarl corpus is usually used. For the transliteration task, common character combinations in the target language may not necessarily be as common in named entities. Hence, we like to test the impact of adding data from a larger set of target language strings to estimate the character language model for our task.

### 7.5.3 Evaluation metrics

We use two metrics to evaluate the translations that are generated. The first measure which is commonly used is accuracy for which we compute the proportion of correctly transliterated names in the test set. Accuracy, as seen from the results in the previous task, is a very strict measure with respect to character-based translation where one single mismatch is counted in the same way as a completely dissimilar pair of strings. Furthermore, for many transliterated names, several alternatives may be acceptable in a language (for example, “Chrushchev” instead of “Khrushchev”) but only one reference is given in our data. In the previous task, we used an F-score measure that is based on the longest common subsequence between the candidate and reference transliterations. For this task, we use the longest common subsequence ratio (LCSR) as our main evaluation measure. Given a pair of strings, LCSR in this case is defined as the ratio of the length of the longest common subsequence and the length of the longer string. LCSR equal to 1 indicates a perfect match between the two strings.

### 7.5.4 Results

Let us first have a look at the baseline for this task. A common technique in machine translation for handling unknown words is to leave them untouched and to copy them to the target output. For names (usually a large portion of unknown words) this is certainly a good strategy if the writing system of the source and target language is very similar. The baseline for our task refers to this strategy of copying the strings even for transliterated names.

Table 7.5 shows the translation results for the WFST models with those for the Baseline at the top. As we can see in Table 7.5, the LCSR baseline scores for both Dutch↔English and French↔English transliteration are quite high already, which means that Dutch and English, or French and English spellings of Russian names...
Applying Pair HMMs in transliteration generation

<table>
<thead>
<tr>
<th>Method</th>
<th>Dut→Eng</th>
<th>Eng→Dut</th>
<th>Fre→Eng</th>
<th>Eng→Fre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.88</td>
<td>0.32</td>
<td>0.88</td>
<td>0.32</td>
</tr>
<tr>
<td>editWFST</td>
<td>0.88</td>
<td>0.22</td>
<td>0.87</td>
<td>0.20</td>
</tr>
<tr>
<td>1 state</td>
<td>0.88</td>
<td>0.22</td>
<td>0.87</td>
<td>0.18</td>
</tr>
<tr>
<td>2 states</td>
<td>0.79</td>
<td>0.00</td>
<td>0.88</td>
<td>0.18</td>
</tr>
<tr>
<td>3 states</td>
<td>0.81</td>
<td>0.12</td>
<td>0.80</td>
<td>0.04</td>
</tr>
<tr>
<td>4 states</td>
<td>0.81</td>
<td>0.06</td>
<td>0.85</td>
<td>0.22</td>
</tr>
<tr>
<td>5 states</td>
<td>0.78</td>
<td>0.02</td>
<td>0.78</td>
<td>0.02</td>
</tr>
<tr>
<td>vow/non-vow</td>
<td>0.83</td>
<td>0.20</td>
<td>0.84</td>
<td>0.28</td>
</tr>
<tr>
<td>phmm9wfst</td>
<td>0.88</td>
<td>0.22</td>
<td>0.87</td>
<td>0.18</td>
</tr>
<tr>
<td>phmm9wfstD+</td>
<td>0.88</td>
<td>0.32</td>
<td>0.88</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Table 7.5: LCSR and accuracy results associated with the use of weighted finite state transducers for character-based translation between English and Dutch, and between English and French. Bolded values indicate better performance over the baseline result.

are not so different from each other. Even the accuracy is also high considering the strict nature of this measure. According to these results, the WFST models do not perform very well. None of the WFSTs actually improves the baseline LCSR nor accuracy for translation between Dutch and English. The translation performed by the WFSTs in this case would harm an SMT system that uses the baseline technique. For French→English, there is a slight improvement in LCSR and accuracy when the edit distance WFST is used. The use of Pair HMM parameters result in a slight improvement only on the LCSR measure for French→English translation. The vowel / non-vowel WFST model performs better than the baseline on the accuracy measure. We can also see that the edit distance WFST does not have a clear advantage over a single-state WFST for translation between the languages in the two language pairs. There is only a slight gain in accuracy for English→Dutch and French→English translation, otherwise the LCSR and ACC values are the same. It is also clear from Table 7.5 that the training procedure is not capable of learning a hidden underlying structure from the data. However, considering the size of our datasets, this should not be expected. Looking at the large differences in the resulting LCSR and ACC for various numbers of states, it seems that the learning algorithm easily gets stuck in suboptimal maxima. Finally, the string splitting strategy of vowel/non-vowel sequences does not improve transliteration generation quality. On the contrary, it actually hurts the model, which is a bit surprising. One reason might be the increased sparseness of our dataset including larger sets of input and output symbols, which now
contain character n-grams. The only improvement associated with the vowel/non-vowel WFST when compared with the other WFSTs can be seen for English→Dutch and French→English translation, albeit only on the accuracy measure. The accuracy for English→Dutch translation, is still below the baseline.

Table 7.6 shows the results from the phrase-based SMT system. We can see a clear improvement in the translation generation quality with regard to the LCSR measure. Except for the non-tuned models with large language models, all LCSR values are above the baseline LCSR value. The importance of training data can be seen in the values for translation between Dutch and English where the tuning set is included in the otherwise very small training data set. For these experiments, we obtain the highest LCSR and accuracy for translation in both directions. For translation between English and French where we have a larger training set, we do not see a similar behavior. A separate development set seems to be preferable. Also, the impact of tuning is mixed and it is not clear how MERT is affected by a setting where the development set is not kept apart from training.

The strategy of splitting characters into vowel/non-vowel sequences makes the PSMT system perform quite well for English→Dutch translation. However, a clear advantage of this strategy over the standard pre-processing technique can not be seen.

<table>
<thead>
<tr>
<th>PSMT</th>
<th>Dut→Eng</th>
<th>Eng→Dut</th>
<th>Fre→Eng</th>
<th>Eng→Fre</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LCSR</td>
<td>ACC</td>
<td>LCSR</td>
<td>ACC</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.88</td>
<td>0.32</td>
<td>0.88</td>
<td>0.32</td>
</tr>
<tr>
<td>without tuning</td>
<td>0.89</td>
<td>0.24</td>
<td>0.90</td>
<td>0.28</td>
</tr>
<tr>
<td>tuned</td>
<td>0.92</td>
<td>0.30</td>
<td>0.90</td>
<td>0.28</td>
</tr>
<tr>
<td>{tune} ⊂ {train}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without tuning</td>
<td>0.91</td>
<td>0.40</td>
<td>0.92</td>
<td>0.32</td>
</tr>
<tr>
<td>tuned</td>
<td>0.93</td>
<td>0.34</td>
<td>0.91</td>
<td>0.40</td>
</tr>
<tr>
<td>vow/non-vow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without tuning</td>
<td>0.90</td>
<td>0.28</td>
<td>0.91</td>
<td>0.48</td>
</tr>
<tr>
<td>tuned</td>
<td>0.89</td>
<td>0.32</td>
<td>0.92</td>
<td>0.44</td>
</tr>
<tr>
<td>large LM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without tuning</td>
<td>0.82</td>
<td>0.06</td>
<td>0.82</td>
<td>0.06</td>
</tr>
<tr>
<td>tuned</td>
<td>0.91</td>
<td>0.26</td>
<td>0.92</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Table 7.6: LCSR and accuracy results associated with the use of a phrase-based statistical machine translation (PSMT) approach for a character-based translation between English and Dutch, and between Dutch and English. The descriptions for the PSMT models can be found in the text. **Bolded** values indicate better performance over the baseline result.
7. Applying Pair HMMs in transliteration generation

Table 7.7: Examples from the Dutch-English test set showing some typical problems of translating transliterations with the models.

<table>
<thead>
<tr>
<th>Dutch input</th>
<th>Correct English</th>
<th>WFST English</th>
<th>PSMT English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrej Tarkovski</td>
<td>Andrei Tarkovsky</td>
<td>Andrey Tarkovski</td>
<td>Andrey Tarkovsky</td>
</tr>
<tr>
<td>Anna Koernikova</td>
<td>Anna Kournikova</td>
<td>Anna Koernikova</td>
<td>Anna Kurnikova</td>
</tr>
<tr>
<td>Aleksandr Solzjenitsyn</td>
<td>Aleksandr Solzhenitsyn</td>
<td>Aleksandr Solzenitsyn</td>
<td>Alexander Solzhenitsyn</td>
</tr>
<tr>
<td>Anton Tsjechov</td>
<td>Anton Chekhov</td>
<td>Anton Tsyekhov</td>
<td>Anton Chechov</td>
</tr>
<tr>
<td>Andrej Sacharov</td>
<td>Andrei Sakharov</td>
<td>Andrey Sakharov</td>
<td>Andrei Sakharov</td>
</tr>
<tr>
<td>Dmitri Sjostakovitsj</td>
<td>Dmitri Shostakovich</td>
<td>Dmitri Syostakovitsy</td>
<td>Dmitri Sjostakovitch</td>
</tr>
<tr>
<td>Leonid Brezjnev</td>
<td>Leonid Brezhnev</td>
<td>Leonid Brezynev</td>
<td>Leonid Bruzhnev</td>
</tr>
</tbody>
</table>

In the final test, we included English, French and Dutch Europarl data (Koehn 2005) for estimating character-based language models (Table 7.6). We can clearly see that the additional data sets harm the translation process and only after tuning does LCSR and accuracy get back to the level of other models using the small language models from the parallel training data. Looking at the weights after tuning, we can also see that the language model weights are very low when using the large datasets. This seems to suggest that the overall influence of a language model on transliteration quality is rather low in our case.

Table 7.7 shows some examples of translations from the Dutch / English test set. In these examples, we can see typical problems especially of the WFST model. In particular, we can see the problem of consistent erroneous character substitutions without considering local context. For example, in the WFST translation, ‘i’ is consistently translated into ‘i’ in English and ‘j’ into ‘y’. For the PSMT model, contextual dependencies are covered better due to the character n-grams in the translation table. However, there are still some ambiguities causing problems like ‘tsjechov’→‘chechov’ (instead of ‘Chekhov’).

7.6 Conclusion

In this chapter, we have used the framework of weighted finite state automata to represent Pair HMMs for transliteration generation. The results associated with English→Russian transliteration generation suggested that the Pair HMM-based WFST models led to better transliteration generation quality compared to the usual application of WFSTs. However, the results also show that the performance from
7.6 Conclusion

all the WFST models reported in this chapter is still lagging behind the state-of-the-art phrase-based statistical machine translation (PSMT) approach. This is obviously attributed to the lack of contextual information in the WFST models as compared to the case for the PSMT approach. On finding that the Pair HMM-based WFST models generated consistent mistransliterations after analyzing results from English→Russian transliteration, the use of some contextual information in the WFST models and the specification of a few contextual transformation rules in a post-processing step resulted in a large improvement in transliteration generation quality, but still below that of the PSMT-based system. The use of additional data also led to improved transliteration generation quality, with a larger improvement associated with the additional modifications to the Pair HMM-based WFST models.

We have also looked at the problem of translating transliterated names between languages that use the same writing system. We again applied WFST and PSMT-based models for transliteration generation between English and Dutch, and between English and French. We trained the models on name pairs of Russian origin extracted from Wikipedia. The PSMT-based approach performed best as expected, consistently beating the baseline of copying strings across the languages. The results in this case show that specialized models like the ones we have tested may help handle ‘unknown’ words in cross-lingual applications between languages using the same writing system when used in a transliteration generation framework.

We have only managed to use Pair HMM parameters in transliteration generation. The limitations that we associated with the Pair HMMs could be captured by the context-dependent transduction-based DBN models, however, we have not yet developed an interface that transforms the transduction-based DBN parameters parameters to a format that is suitable for use in transliteration generation. The results suggested improved transliteration generation quality in regard to using contextual information in the Pair HMM - based WFSTs, it would be interesting to determine whether parameters from context-dependent DBN models (Chapter 5) could result in improved transliteration generation quality.
Chapter 8

Conclusions and future work

8.1 Conclusions

In this thesis, several Dynamic Bayesian Network (DBN) models have been presented and applied in transliteration mining and generation. The need to investigate the application of the DBN models in transliteration mining and generation is partly rooted in the observation of their success in NLP tasks with requirements similar to transliteration mining. This observation resulted in the proposal to apply two DBN approaches in transliteration mining and generation with a goal of determining whether the DBN models could improve transliteration mining and transliteration generation quality over state-of-the-art approaches. In the following, we present following the outline of the thesis presented in Chapter 1 a summary of the different stages of our investigation and the related contributions.

Since there are many methods for both transliteration mining and generation, we have first of all, provided a more recent and exhaustive review on transliteration mining and generation while aimed at showing the general setup per task, identifying various methods that have been applied in the two tasks ranging from the earliest to current state-of-the-art, and establishing the necessity to apply proposed DBN approaches. The literature review addresses the first research question where we wanted to know whether existing methods for transliteration generation and mining suffice. As answers to the first research question, we found out the following. One, that because of the large amount of research on transliteration mining and generation, a number of factors pertinent to transliteration had been addressed in the various methods. However, the reports on the recent shared tasks on transliteration mining and generation showed that state-of-the-art approaches are still far from achieving high quality results as reflected in system evaluations for the two tasks. In other words, there is still need to improve transliteration mining and generation quality. Secondly, we found out that some DBN models (especially in the form of the classic Hidden Markov models (HMMs)) had already been used in transliteration mining.
and generation. However, to the best of our knowledge, there existed no report with regard to the use of the DBN approaches proposed in the thesis for transliteration mining and generation.

For the second research question, we wanted to know whether DBN models that had been used in tasks with requirements similar to transliteration mining and modifications of the DBN models that meet requirements for computing transliteration similarity could be valuable in the identification of transliteration pairs. In addressing this question, we have presented a conceptual framework for the Pair HMM approach in Chapter 4 and for the transduction-based DBN approach in Chapter 5 describing the adaptation of different DBN models to compute transliteration similarity. Based on the ideas presented there, we undertook an empirical investigation (in Chapters 4 and 5) into the use of various parameterizations of the DBN models in a transliteration identification task using standard transliteration datasets. With reference to the second research question, results show that the DBN models achieve considerable accuracy gains if they are designed to specifically represent source and target writing systems. Chapter 4 results showed that the Pair HMMs have a lower cross entropy (less uncertainty) with respect to a given transliteration corpus if they use two alphabets (corresponding to the source and target writing systems) to explain the relationship between source and target words than when they use one ‘universal’ alphabet. This result suggests that the assumption of Pair HMMs using one alphabet in the cognate identification and dialect comparison tasks can not be relied on in computing transliteration similarity. In Chapter 4, we also investigated four Pair HMM settings with differences in the definitions of transition parameters. Results showed that it is important to include all standard transition parameters when computing transliteration similarity. However, in models using log-odds ratio to compute transliteration similarity, the results suggested no guarantee of improving transliteration identification quality with an increase in the number of transition parameters since the performance was almost similar for all Pair HMMs where transition parameters were used. In Chapter 5, we evaluated four edit-distance based DBN model generalizations in the same transliteration identification setup used for the Pair HMM variants. Chapter 5 results showed that context-dependent DBN models achieve better transliteration identification quality compared to other DBN models. The evaluation of the different DBN models also addresses the third research question where the aim was to identify DBN models that adequately model transliteration similarity. For this question, Chapter 4 results show that the use of transition parameters and the involvement of a random model when using Pair HMMs to compute transliteration similarity results in a more superior and stable performance than when a random model is not used. Chapter 5 reports how Pair HMMs were compared against the edit-distance based DBN models and it shows that
context-dependent DBN models achieve better transliteration identification quality for at least 5/7 language pairs. The success of the context-dependent DBN models here underlines the necessity to model character context for computing transliteration similarity.

In addressing the second and third research questions for the transliteration generation task, we also provided a conceptual framework for representing DBN models as weighted finite state automata to enable an evaluation of the use of their parameters in the task. We defined different weighted finite state transducers (WFSTs) as approximations to Pair HMMs. The WFSTs are used to encode corresponding Pair HMM parameters. The Pair HMM-based WFST models are, however, disadvantaged by the amount of source and target character contextual information they model compared to state-of-the-art phrase-based statistical machine translation (PSMT) approaches (Matthews 2007, Finch and Sumita 2008) and approaches that first induce many-to-many alignments (Jiampojamarn et al. 2010). An evaluation of the Pair HMM-based WFSTs in Chapter 7 on standard transliteration corpora for seven language pairs shows that the Pair HMM-based WFSTs result in weaker transliteration generation quality compared to that for the PSMT and many-to-many alignment approaches. However, an empirical investigation into the representation of some context in the Pair HMM-based WFSTs results in a greater improvement in transliteration generation quality on an English→Russian dataset compared to the case when context is not represented (Nabende 2009).

For the last research question, we wanted to know whether the application of DBN models could improve transliteration mining and generation quality compared to state-of-the-art methods. To address this question, we first evaluated the two DBN approaches against state-of-the-art approaches in mining transliterations from standard Wikipedia paired topics (also referred to as Wikipedia inter-language links (WILs)) which were provided as standard corpora for evaluating language-independent systems in the NEWS 2010 shared tasks on transliteration mining (Kumaran et al. 2010b). The results in Chapter 6 suggest a performance comparable to that for state-of-the-art methods (Nabende 2010). Chapter 6 results also show a considerable improvement in transliteration mining quality from using Pair HMMs over state-of-the-art approaches on transliteration corpora for two language pairs: English-Hindi and English-Tamil. In addition to using only Wikipedia paired topics, we also proposed to apply the DBN models in mining transliterations from the main content of comparable, bilingual Wikipedia pages. Based on the premise that the number of words contained in the article content exceed the number of words in the topics by a multiple, we expected increased coverage by extracting transliteration pairs from the article content. Research on mining transliterations from Wikipedia commonly involves the use of training data that is prepared from an external source. In our
8. Conclusions and future work

In this case, we also show that it is possible to use only Wikipedia data for mining transliterations. Specifically, we have applied a method from related work where we search for only particular types of Wikipedia topics such as person names which are in most cases standard translations across languages and used them as training data. Results suggest a promising application of the DBN models in mining transliterations from the very ‘noisy’ comparable Wikipedia article content (Nabende 2011).

Finally, previous work on transliteration mining and generation emphasizes mapping from one writing system to another. However, cross-language processing between languages that use the same writing system, with the presence of unknown entities, is affected in a manner similar to the case where the languages use different writing systems. We therefore proposed the usual application of the transliteration mining and generation framework to that where the languages use the same writing system. To the best of our knowledge, this task had not yet been addressed in transliteration mining and generation literature. After testing various models, we show that the usual transliteration generation setup leads to considerable accuracy gains over the standard baseline of copying strings from the source language to the target language (Tiedemann and Nabende 2009).

8.2 Future work

Although our conclusions show a valuable application of DBN models in transliteration mining, a number of interesting research directions can follow from the work presented in the thesis. First of all, we begin with what is unfinished.

For the transliteration mining task, we proposed an investigation into several DBN model settings. But as is described in the thesis, the DBN approaches offer a limitless model space for which we have not exhaustively explored. For the edit distance based DBN modeling approach in particular, we propose an investigation into the application of additional models. We also evaluated the DBN models against the state-of-the-art methods on only three language pairs. It should be interesting to evaluate their performance on real-world data for additional language pairs.

For the transliteration generation task, we have only investigated the use of Pair HMM parameters in transliteration generation. One interesting research direction is to investigate the use of the parameters for the edit distance based DBN models in the transliteration generation task. Since the edit distance based DBN models are based on a representation of a stochastic memoryless transducer, we postulate that a transformation of the DBN models to finite state automata representations should be possible so as to enable the evaluation of the use of DBN model parameters for transliteration generation.

There are other research directions that can be followed in addition to our unfinished...
lished work above. Here, we would like to note here that our investigation into the use of the edit distance based DBN models was mostly affected by processing speed during inference based on the models. We have reported in the use of the Frontier algorithm, but there are other algorithms that have been proposed to improve computational efficiency while maintaining and in some cases improving effectiveness in using DBN models. It should be interesting to investigate the use of other inference algorithms for the edit-distance based DBN models.

Our work is just but an additional application of the two DBN approaches in machine transliteration. Our successful application of the DBN models in mining transliterations from noisy Wikipedia data suggests the application of the DBN models to problems where there is need to handle ‘noise’ in sequences. It should also be interesting to investigate the application of the DBN approaches presented in this thesis to address a variety of problems based on edit operations.


Li, H., Zhang, M. and Su, J.: 2004, A joint source-channel model for machine transliteration, 
*Proceedings of the 42nd meeting of the Association for Computational Linguistics (ACL '04),* Barcelona, Spain, pp. 159–166.


Bibliography


