Dynamic Bayesian Networks for Transliteration
Discovery and Generation

Peter Nabende
Alfa Informatica, CLCG,
University of Groningen
p.nabende@rug.nl

May, 2009
Contents
Introduction ..................................................................................................................................... 1
  1.1 Background ............................................................................................................................. 1
  1.2 Definitions ............................................................................................................................... 2
  1.3 Motivation ............................................................................................................................... 3
  1.4 Research Questions ................................................................................................................... 4
  1.4 Research Objectives .................................................................................................................. 5
  1.5 Overview ................................................................................................................................... 5

Literature Review: Transliteration Discovery and Generation ....................................................... 6
  2.1 Transliteration Discovery ........................................................................................................... 6
  2.1 Transliteration Generation ......................................................................................................... 8

Dynamic Bayesian Networks ........................................................................................................ 10
  3.1 Introduction ............................................................................................................................... 10
  3.2 Graphical Models ..................................................................................................................... 10
    3.2.1 Types of Graphical Models .......................................................................................... 11
  3.3 Bayesian Networks ................................................................................................................... 12
    3.3.1 Types of Bayesian Networks ..................................................................................... 12
    3.3.2 Inference in Bayesian Networks ................................................................................. 12
    3.3.3 Learning in Bayesian Networks ................................................................................. 13
  3.4 Dynamic Bayesian Networks .................................................................................................. 14

Identification of bilingual entity names using a pair HMM ............................................................. 17
  4.1 Introduction ............................................................................................................................... 17
  4.2 Hidden Markov Models ........................................................................................................... 17
  4.3 Pair-Hidden Markov Model ..................................................................................................... 19
    4.3.1 Proposed modifications to the parameters of the pair HMM ....................................... 21
  4.4 Application of pair HMM to bilingual entity name identification ........................................... 22
    4.4.1 Parameter Estimation in pair HMMs ........................................................................... 23
    4.4.2 Baum-Welch Algorithm for the pair HMM ................................................................. 23
    4.4.3 pair-HMM Training Software .................................................................................... 28
  4.5 Experiments ............................................................................................................................ 28
    4.5.1 Forward pair HMM algorithm vs. Viterbi pair HMM algorithm .................................. 29
      4.5.1.1 Training data and Training time ........................................................................... 30
Submitted............................................................................................................................................ 62
Conference Presentations.................................................................................................................... 62
Poster Presentations ............................................................................................................................ 63
Research Talks .................................................................................................................................... 63
Appendix B: PhD Research schedule ................................................................................................. 64
Chapter 1

Introduction

1.1 Background

This project is involved with extraction and transliteration of entity names between languages that use different writing systems (or alphabets). Extraction involves the automatic identification of sequences in parallel/comparable corpora/text that can be considered as proper entity names. On the other hand, transliteration generation involves automatic transformation of a source language name to a target language name across different writing systems while ensuring that pronunciation is maintained during and after the process. Bilingual entity name recognition and/or transliteration are processes aimed at improving performance in various Natural Language Processing (NLP) applications including: Machine Translation (MT), Cross Language Information Retrieval (CLIR), and Cross Language Information Extraction (CLIE). As an example, in recognition, there has been a growing interest in recognizing variations for the same entity name across different languages (Hsu et al., 2007; Pouliquen et al., 2006; Kondrak and Dorr, 2004). Table 1.1 illustrates variations for the name of the Russian President (sworn in on 7th May 2008) in both the source language that uses the Cyrillic alphabet and target languages that use the Roman alphabet. The presence of variations for the same entity may lead to incomplete search results and cause communication problems in a larger community of language users (Hsu et al., 2007). In medication, identification of drug names that look similar has been

<table>
<thead>
<tr>
<th>Source in Russian that uses the Cyrillic alphabet</th>
<th>Transliterations in languages that use the Roman alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Дмитрий Медведев</td>
<td>Дмитрий Медведев</td>
</tr>
<tr>
<td>Дмитрий Медведев</td>
<td>Дмитрий Медведев</td>
</tr>
<tr>
<td>Дмитрий Медведев</td>
<td>Дмитрий Медведев</td>
</tr>
<tr>
<td>Дмитрий Медведев</td>
<td>Дмитрий Медведев</td>
</tr>
</tbody>
</table>

Table 1.1 Example illustrating variations for the same name both its source language and languages to which it was transliterated (Source: NewsExplorer Website1)

---

1 NewsExplorer website can be accessed via http://press.jrc.it/NewsExplorer/entities/en/48520.html
Mbale is a town on the slopes of Mountain Elgon in Eastern Uganda

Table 1.2: Example illustrating translation of a phrase with an Out of Vocabulary word found to be very helpful in reducing drug prescription errors (Kondrak and Dorr, 2004). To stress the need for machine transliteration, table 1.2 shows a problem where a machine translation system (Google\textsuperscript{2} translate engine) encounters a new entity name “Elgon” in English for which it can not translate in a target language (Russian). Such a situation arises because the translation system does not have this word in its translation dictionary or lexicon. Machine transliteration is one of the best approaches to help deal with Out Of Vocabulary (OOV) problems.

This report proposes work in the framework of Dynamic Bayesian Networks (DBNs) in which various model spaces are investigated with the aim of improving entity name recognition and machine transliteration across different writing systems. The concepts associated with DBNs are introduced and for experimental work, we start by investigating the simplest class of DBN models called Hidden Markov Models (HMMs). In particular, a pair Hidden Markov Model is adapted for application to estimating similarity between candidate transliteration pairs, and to providing parameters for generating transliterations between two languages. The performance of the pair HMM is evaluated against models based on Weighted Finite State Transducers (WFSTs) and on state of the art Phrase-Based Statistical Machine Translation (PBSMT) techniques. Apart from DBNs, various Finite State Automatons are evaluated for translating transliterations whose origins are in a language using a different writing system.

1.2 Definitions

The definitions for some of the terms below are provided within the context of this report. It is possible that these terms may have different definitions in other contexts.

Entity Names

Named entities are generally divided into three types (Chinchor, 1997): entity names, temporal expressions, and number expressions. In the transliteration extraction and generation tasks presented in this report only entity names are considered, that is: organization, person, and

\textsuperscript{2} http://translate.google.com
location names. Henceforth, the terms “Named Entities” and “Entity Names” will be used interchangeably.

**Bilingual named entity recognition.**

Bilingual named entity recognition refers to the process of determining the similarity between candidate named entities across different languages with the aim of extracting the most similar target name(s) for a given source name. In this report, the focus is on extracting matching named entities from a set of candidate named entities for languages that use different writing systems such as English (Roman alphabet) and Russian (Cyrillic alphabet).

**Machine Translation**

Machine Translation (MT) refers to the use of computer software to translate text from one natural language to another natural language. MT usually involves: breaking down sentences, analyzing them in context, and recreating their meaning in the target natural language. In this report, various models based on WFSTs and SMTs are investigated for translating entity names between two languages. The machine translation task in this case involves translating characters in an entity name instead of translating words in a sentence.

**Machine Transliteration**

Transliteration is defined as the process of transcribing a word written in one writing system into another writing system while ensuring that the pronunciation is as close as possible to the original word or phrase. Forward transliteration converts an original word or phrase in the source language into a word in the target language, whereas backward transliteration is the reverse process that converts the transliterated word or phrase back into its original word or phrase (Lee et al., 2003). In this report, the term transliteration is associated with two contexts: one context refers to the transliteration process while the other context refers to the outcome from a transliteration process. Otherwise, Machine Transliteration refers to the automation of the process of transliteration (Oh et al., 2006).

**1.3 Motivation**

Automatic identification of bilingual entity names and transliteration generation for languages using different writing systems aids in major NLP applications such as MT, CLIE and CLIR by
increasing coverage of entity name representations and increasing quality of output from the NLP applications. To obtain good performance for the NLP applications, there is need to use efficient and portable models for both the recognition and generation tasks. Statistical approaches provide most of the current state-of-the-art methods and techniques and have been applied with significantly good performance in various NLP applications. One framework that is appealing for exploring various model spaces for bilingual named entity recognition and transliteration generation is that of Dynamic Bayesian Networks (DBNs). Some DBN models have been investigated on tasks similar to the bilingual named entity similarity estimation task in this report and were performed relatively well on those tasks (Filali and Bilmes, 2005; Kondrak and Sherif, 2006). Moreover some of the DBN models investigated in previous work address issues such as context and memory that are very important for the recognition and transliteration generation tasks introduced in this report.

1.4 Research Questions

(1) What are the main requirements for representing transliteration extraction and transliteration generation problems?

- To answer this question, a literature review on transliteration extraction and generation is needed. From the literature review, it is necessary to identify state of the art techniques for extracting bilingual named entities and those used for transliteration generation with the aim of identifying differences and limitations against models that are proposed for the transliteration tasks in this project.

(2) How can the framework of Dynamic Bayesian Networks be applied for bilingual entity name recognition and transliteration generation?

- This question can be answered through: a clear introduction of concepts associated with Dynamic Bayesian Networks (DBNs) and specifying the main features that make them suitable for transliteration tasks. Also needed is a clear specification of approaches for applying DBNs for transliteration tasks.

(3) What level of transliteration quality can be achieved through use of Dynamic Bayesian Network models on a transliteration tasks?
- To be able to determine how the proposed DBN models perform, various quality measures can be used that are associated with the two transliteration tasks. The metrics serve to evaluate the proposed DBN models against other methods for the two transliteration tasks.

1.4 Research Objectives

• Main Objective
  – To adapt and develop DBN models for recognizing and generating entity names across different languages and writing systems with the main goal of achieving improvements in the performance of various NLP applications for example MT and CLIR systems.

• Specific Objectives
  – In the recognition of entity names, I wish to achieve the objective of adapting and developing DBN models to estimate the similarity between entity names from different languages that use either the same writing system or different writing systems. The similarity estimations can then be used for different recognition and translation applications.
  – In generating entity names across different languages and writing systems, I wish to adapt and develop statistical translation and transliteration models with the aim of improving performance in MT and CLIR applications.
  – To evaluate DBN techniques and methods proposed above against different techniques and methods for example rule-based techniques and methods that are currently used for named entity recognition and machine translation. The aim here is to determine limitations in the statistical methods that can be solved either by using different methods or in combination with different methods.

1.5 Overview

The rest of the report describes work that has been done with regard to the NLP tasks associated with my research: In chapter 2, concepts associated with the DBN framework are introduced, and in chapter 3, the name recognition task where a pair HMM is adapted is described; machine translation and transliteration tasks are described in Chapter 4. For each of the tasks: the techniques, methods, models, and systems adapted are described including the experiments carried out. Information about publications, research talks, and courses attended with regard to the work reported in this report can be found in the Appendices.
Chapter 2

Literature Review: Transliteration Discovery and Generation

In this chapter, a literature review is given with regard to the two transliteration tasks for which the DBN framework is proposed: extraction of transliteration pairs from bilingual texts, and automatic generation of a transliteration from a source language entity name.

2.1 Transliteration Discovery

The task here is to automatically identify a set of potential target language entity names for a set of source language entity names. Any method that is used to automatically discover transliterations between two languages will require a bilingual text corpus. Bilingual text corpora can be divided into three types (Mcenery and Xiao, 2007): Source texts plus translations (Type A), Monolingual sub corpora (Type B), a combination of Type A and Type B (Type C). However, as shown by Mcenery and Xiao (2007) there exists confusion surrounding the terminology used in relation to the different types of corpora. The term parallel corpus is used here to refer to Type A, while the term comparable corpora is used to refer to any bilingual text corpora that is of Type B or Type C. Extraction of bilingual named entities from parallel corpora follows three steps (Erdmann, 2008): 1) corpus preparation, 2) sentence alignment, and 3) named entity matching. For the case of comparable corpora, extraction of bilingual named entities is in two main steps: 1) corpus preparation and 2) named entity matching. The methods used in the step of matching named entities are similar for both types of bilingual text corpora. The general strategies used for named entity matching are the same for both types of corpora (Lee et al., 2006; Moore, 2003): asymmetric strategy which assumes that NEs in the source language are given and the task is to identify matching translations in the target language; and symmetric strategy which tries to identify NEs in the source and target language and then establish a relationship between the NE pairs. Based on the two strategies various techniques have been used. Some recent techniques are described below.
Lam et al. (2007) argue that many named entity translations involve both semantic and phonetic information at the same time. Lam et al. (2007) hence developed a named entity matching model that offers a unified framework for considering semantic and phonetic clues simultaneously in estimating the similarity between bilingual named entities. They formulate the named entity matching problem via an undirected bipartite weighted graph. In the bipartite graph, each vertex is associated with a token and each edge is associated with a semantic or a phonetic mapping between an English token and a Chinese token. Each edge has a weight that is determined by the degree of mapping of associated tokens (Lam et al., 2007). They then find a set of edges so that the total weight is maximized and each token can only be mapped to a single token on each side of the bipartite graph.

Hsu et al. (2007) measure the similarity between two transliterations by comparing the physical sounds through a Character Sound Comparison (CSC) method. The CSC method is divided into two parts: the first stage is a training stage where a speech sound similarity database is constructed that includes two similarity matrices; the second stage is a recognition stage where transliterations are mapped to a phonetic notation and the similarity matrices from the training stage are used to calculate the phonetic similarity.

Pouliquen et al. (2006) compute the similarity between pairs of names by taking the average of three similarity measures. Their similarity measures are based on letter n-gram similarity. They calculate the cosine of the letter n-gram frequency lists for both names, separately for bi-grams and for tri-grams; the third measure being the cosine of bigrams based on strings without vowels. Pouliquen et al. (2006) do not use phonetic transliterations of names as they consider them to be less useful than orthographically based approaches. Because of various limiting factors, however, Pouliquen et al. (2005) obtain results that are less satisfactory than for language-specific Named Entity Recognition systems. The precision obtained from their results was, nevertheless higher.


2.1 Transliteration Generation

Two types of transliteration generation exist: Forward Transliteration where a word in a source language is transformed into target language approximations; and Backward Transliteration where target language approximations are transformed back to an original source language word. In either direction, the transliteration generation task is to take a character string in one language as input and automatically generate a character string in the other language as output. The transliteration generation process usually involves segmentation of the source string into transliteration units; and associating the source language transliteration units with units in the target language by resolving different combinations of alignments and unit mappings (Li et al., 2004). The transliteration units usually comprise of a phonetic representation, a Romanized representation or original orthographic (grapheme) representation. Three major types of models can be identified with regard to the type of transliteration units used (Oh et al., 2006): grapheme-based models, phoneme-based models, and hybrid models that are a combination of grapheme and phoneme models. Across all types of models used, three major classes of algorithms are common (Zhang et al., 2004): rule-based methods and statistical methods combined with machine learning techniques, and a combination of both. The earlier approaches to machine transliteration were mainly rule-based methods (Arbabi et al., 1994; Yamron et al., 1994; Wan and Verspoor, 1998). However, it is difficult to develop rule-based systems for each language pair. The first major reported work utilizing statistical and machine learning techniques was by Knight and Graehl (1997). Knight and Graehl (1997) used generative techniques relying on probabilities and Bayes’ rule. The five probability distributions that Knight and Graehl considered for phoneme-based transliteration between English and Japanese Katakana were implemented using Weighted Finite State acceptors (WFSA) and transducers (WFSTs). Some of the WFSTs used by Knight and Graehl (1997) were learned automatically using the Expectation Maximization (EM) algorithm (Baum, 1972) while others were generated manually. Related work adapting the techniques used by Knight and Graehl (1998) include: Arabic-English back transliteration (Stalls and Knight, 1998), Arabic-English spelling-based transliteration (Stalls and Knight, 1998; Al-Onaizan and Knight, 2002). Several other generative models were proposed to improve on transliteration generation quality. These include: source channel model (Lee and Choi, 1998), extended markov window (Jung et al., 2000), joint source channel model (Li et al.,
Apart from generative methods, discriminative methods have also been used for machine transliteration. Oh and Isahara (2007) use MaxEnt models for representing probability distributions used in transliteration between English and Arabic. Zelenko and Aone (2006), use two discriminative methods that correspond to local and global modeling. In the local setting, Zelenko and Aone (2006) learn linear classifiers that predict a target transliteration unit from previously predicted transliteration units. In the global setting, Zelenko and Aone learn a function that maps a pair of transliteration units into a score, and the function is linear in features computed from the pair of transliteration units. Klementiev and Roth use a linear model to decide whether a target language word is a transliteration of a source language word.

There are also approaches that combine generative and discriminative models, most especially using discriminative training for generative models.

Most of the approaches described above try to improve machine transliteration quality by investigating different issues. In this report, the framework of Dynamic Bayesian Networks is proposed for investigation for both transliteration generation and extraction. DBNs models that have been used before in NLP are mostly generative models. However, there is chance to use discriminative training procedures for a given DBN model. Among the issues of interest that should be modeled by DBNs include: contextual and history information while transforming a source language word to a target language transliteration.
Chapter 3

Dynamic Bayesian Networks

3.1 Introduction

Dynamic Bayesian networks (DBNs) are directed graphical models that model sequential or time series problems. DBNs have been applied before for NLP. Deviren et al. (2004) use DBNs for language model construction; Filali and Bilmes (2005) investigate various DBN models for word similarity estimation; and Kondrak and Sherif (2006) test the DBNs proposed by Filali and Bilmes (2005) for cognate identification; Filali and Bilmes (2007) also propose Multi-Dynamic Bayesian Networks (MDBNs) for machine translation word alignment. This chapter introduces the concepts associated with DBNs, but first the general framework of Graphical models from which DBNs are derived is introduced.

3.2 Graphical Models

Graphical models (GMs) are basically a way of representing probability distributions graphically. They combine ideas from statistics, graph theory and computer science. GMs simplify the representation and reading of conditional independencies which play an important role in probability reasoning and machine learning. Hence, GMs allow definition of general algorithms that perform probabilistic inference efficiently. GMs consist of a set of random variables (nodes) denoted here as $X = \{x_1, x_2, \ldots, x_k\}$, a set $E$ of dependencies (edges) between the variables, and a set $P$ of probability distribution functions for each variable. Two variables $x_1$ and $x_2$ are independent when they have no direct impact on each other’s value (Figure 3.1a). In figure 3.1b, variables $x_1$ and $x_2$ become conditionally independent given a third variable $x_3$ on which $x_1$

![Figure 3.1](image_url)
and $x_2$ depend. This conditional independence can be expressed as: $P(x_1 \mid x_3, x_2) = P(x_1 \mid x_3)$.

### 3.2.1 Types of Graphical Models

Graphical models can be divided into two major types: Undirected graphical models and Directed graphical models. As the names suggest, undirected graphical models are such that there is no directionality attached to any of the edges in the graph. Whereas in directed graphical models, directionality is explicitly shown at the end of each edge in the graph showing directionality from one node to another. While undirected graphs express soft constraints between random variables, directed graphical models express causal relationships between random variables. For purposes of solving inference problems, it is often convenient to convert both directed and undirected graphs into *factor graphs* (Bishop, 2006). Figure 3.2 shows the two major types of graphical models with examples. DBNs are classified under Bayesian Networks (BNs) which are directed GMs. Bayesian Networks are introduced in the next section.

![Figure 3.2: Classification showing examples of graphical models](image-url)
3.3 Bayesian Networks

A Bayesian Network is a specific type of graphical model which is a Directed Acyclic Graph (DAG). A DAG is a graph in which all the edges in the graph are directed and there are no cycles. Directionality indicates parent \((x_p)\) to child \((x_c)\) or cause to effect relationship (figure 3.3).

![Figure 3.3: Bayesian Network showing causal relationship between parent and child variables](image)

A Bayesian network compactly represents a joint probability distribution over a set of variables \(X\). For the BN of figure 3.3, the expression for the joint probability is:

\[
P(x_{c1}, x_p, x_{c2}) = P(x_{c1} \mid x_p) \cdot P(x_p) \cdot P(x_{c2} \mid x_p)
\]

In general, given a set of nodes \(X = \{x_1, ..., x_n\}\) in a BN, the joint probability function is given as:

\[
P(X) = \prod_{i=1}^{n} P(x_i \mid \text{parents}(x_i))
\]

The graph serves as a backbone for efficiently computing the marginal and conditional probabilities that may be required for inference and learning.

3.3.1 Types of Bayesian Networks

Depending on the problem, different types of BNs can be created. Two main classes exist: singly connected (Figure 3.4 (a)) and multiply connected (Figure 3.4 (b)) BNs. Singly connected networks are those for which only one path exists between any two nodes in the network, while multiply connected are those for which there is more than one path for some two nodes in the network. Depending on the type of network, different inference methods can be used.

3.3.2 Inference in Bayesian Networks

Two major types of inference are possible: exact inference and approximate inference. Exact inference involves a full summation (integration) over discrete (continuous) variables and is NP
The most common exact inference methods include: Variable Elimination and message passing algorithm. For some problems with many variables, exact inference techniques may be slow. Instead approximate inference techniques have to be used. Major approximate inference techniques include (Murphy, 1998): variational methods, sampling methods, loopy belief propagation, bounded cutest conditioning, parametric approximation methods.

### 3.3.3 Learning in Bayesian Networks

The role of learning is to adjust the parameters of a Bayesian network so that the probability distributions defined by the network sufficiently describe the statistical behavior of the observed data. Generally, there are four Bayesian Network learning cases that are often considered, and to which different learning methods have been proposed as shown in the table 3.1.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Observability</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known</td>
<td>Full</td>
<td>Sample statistics</td>
</tr>
<tr>
<td>Known</td>
<td>Partial</td>
<td>EM or gradient ascent</td>
</tr>
<tr>
<td>Unknown</td>
<td>Full</td>
<td>Search through model space</td>
</tr>
<tr>
<td>Unknown</td>
<td>Partial</td>
<td>Structural EM</td>
</tr>
</tbody>
</table>

Table 3.1: Learning methods for BNs depending on what is already known about the problem

(Source: Murphy and Mian, 1999)
3.4 Dynamic Bayesian Networks

Dynamic Bayesian Networks (DBNs) are Bayesian Networks in which variables have a relation to time.

To specify a Dynamic Bayesian Network (DBN), the following needs to be defined (figure 3.5): a) a prior network, b) a Transition network, c) an observation network, and d) an end network.

Fig 3.5 Networks needed to completely specify a DBN for the simple example of the classical HMM

Figure 3.6 and 3.7 show the network structure for the classical HMM and the pair HMM respectively. Classical HMMs have only one hidden variable and observation variable, while pair HMMs have one hidden variable and two observation variables. Dynamic Bayesian Networks generally have any number of hidden variables and any number of observation variables.

Fig 3.6 A HMM, a simple type of DBN

Fig 3.7 pair HMM with two observations
Dynamic Bayesian Networks have several advantages when applied to machine transliteration tasks. One major advantage is that, complex dependencies associated with different factors such as context, memory, and position in strings involved in a transliteration process can be captured. The challenge then is to specify DBN models that naturally represent the transliteration problem while addressing some of the factors. One suitable approach that can be adapted from previous work is based on estimating string edit distance through learned edit costs (Mackay and Kondrak, 2005; Filali and Bilmes, 2005). As is the case in (Filali and Bilmes, 2005), it is quite natural to construct DBN models representing additional dependencies in the data which are aimed at incorporating more analytical information. In the following DBN models introduced by Filali and Bilmes for computing word similarity are introduced. Figure 3.8 shows a baseline DBN used by Filali and Bilmes (2005), which is referred to as the Memoriless Context Independent (MCI) DBN.

Figure 3.8 The MCI model (Source: Filali and Bilmes, 2005)
In the MCI model, $Z$ denotes the current edit operation, which can be a substitution, an insertion, or a deletion. The lack of memory signifies that the probability of $Z$ taking on a given value does not depend in any way on what previous values of $Z$ have been. The context-independence refers to the fact that the probability of $Z$ taking on a certain value does not depend on the letters of the source or target word. The $a$ and $b$ nodes represent the current position in the source and target strings, respectively. The $s$ and $t$ nodes represent the current letter in the source and target strings. The end node is a switching parent of $Z$ and is triggered when the values of the $a$ and $b$ nodes move past the end of both the source and target strings. The $sc$ and $tc$ nodes are consistency nodes which ensure that the current edit operation is consistent with the current symbols in the source and target strings. Consistency, means that the source side of the edit operation must either match the current source symbol or be $\epsilon$ and that the same is true for the target side. Finally, the send and tend nodes appear only in the last frame of the model, and are only given a positive probability if both words have already been completely processed, or if the final edit operation will conclude both strings.

Additional DBN structures proposed by Filali and Bilmes (2005) use the MCI model as a basic framework, while adding new dependencies to $Z$. As examples: in the context-dependent model (CON), the probability that $Z$ takes on certain values is dependent on symbols in the source string or target string; while in the memory model (MEM), the probability of the current edit operation being performed depends on what the previous operation was.

Given a DBN model, inference and learning will involve computing posterior distributions. Fortunately, there exist efficient, generic exact or approximate algorithms that can be adopted for inference and learning.
Chapter 4
Identification of bilingual entity names using a pair HMM

4.1 Introduction

Identification of bilingual entity names mainly involves determining the similarity between entity names from languages with the same writing system (for example English and French) or languages with different writing systems (for example English and Russian). We find that estimating the similarity between entity names for languages that use different writing systems is relatively difficult compared to similarity estimation when the languages use the same writing system. In this chapter, a pair Hidden Markov Model (pair HMM) is adapted for estimating the similarity between candidate bilingual entity names for languages that use different writing systems. The pair HMM is later tested for extracting transliteration pairs from English and Russian Wikipedia data dumps.

4.2 Hidden Markov Models

The first model we adapt and investigate for estimating the similarity between bilingual entity names belongs to a family of statistical models referred to as Hidden Markov Models (HMMs). HMMs in general are one of the most important machine learning models in NLP (Jurafsky and Martin, 2009). HMMs are also based on a solid statistical foundation with the property of enabling efficient training and application (Baldi and Brunak, 2001).

Rabiner (1989) described how the concept of Markov models can be extended to include the case where the observation is a probabilistic function of the state, i.e., the resulting model (a HMM), is a doubly embedded stochastic process with an underlying stochastic process that is not observable. Figure 4.1 illustrates an instantiation of a HMM.
Formally, an HMM $\mu$ is defined as a tuple:

$$\mu = (A, E, \pi)$$

This definition assumes a state alphabet set $S$ and an observation alphabet set $V$:

$$S = \{ s_1, s_2, \ldots, s_N \}, \quad V = \{ v_1, v_2, \ldots, v_M \}$$

$Q$ is defined to be a fixed state sequence of length $T$, with corresponding observations $O$:

$$Q = q_1, q_2, \ldots, q_T, \quad O = o_1, o_2, \ldots, o_T \quad \text{where each } q_i \in S, \text{ and each } o_t \in V \text{ for } t = 1, \ldots, T.$$  

We also have $A$, a state transition matrix having transition probabilities between all states; transition probabilities are independent of time:

$$A = [p_{ij}], \quad p_{ij} = P(q_{i+1} = s_j | q_i = s_i).$$

$E$ is the observation matrix, storing the probability of observation $v_k$ being produced from the state $s_j$, independent of time $t$:

$$E = [e_j(v_k)], \quad e_j(v_k) = P(o_{t+1} = v_k | q_{t+1} = s_j).$$

$\pi$ is the initial probability matrix:

$$\pi = [\pi_i], \quad \pi_i = P(q_1 = s_i).$$

Two major assumptions are made for a HMM. The first assumption is referred to as the Markov Assumption and is specified as:

$$\forall t \leq n \text{ and } q \in S: P(q_t = s_j | q_{t-1} = s_i, \ldots, q_1 = s_e) = P(q_t = s_j | q_{t-1} = s_i) = p_{ij} \quad (4.1)$$

The independence assumption states that the observation output at time $t$ is dependent only on the current state; it is independent of previous observations and states:

$$P(o_t | q_{t-1}, q_t') = P(o_t | q_t) \quad (4.2)$$
4.3 Pair-Hidden Markov Model

Durbin et al. (1998) made changes to a Finite State Automaton and converted it into an HMM which they called a pair HMM. The main difference between a pair HMM and a classic HMM lies in the observation of a pair of sequences (Figure 3.2) or a pairwise alignment instead of a single sequence. The pair HMM also assumes that the hidden (that is the non-observed alignment sequence \( q_t \)) is a Markov chain that determines the probability distribution of the observations (Arribas-Gil et al., 2005). Transition and emission probabilities define the probability of each aligned pair of strings. Given two input sequences, the goal is to compute the probability for all alignments associated with the pair of sequences or to look for an alignment of the two sequences that leads to maximum probability.

![An instantiation of a pair Hidden Markov Model](image)

For experimental work, we adapt the pair HMM toolkit used by Wieling et al. (2007) for Dutch dialect comparison; but the structure of the pair HMM remains the same as initially introduced by Mackay and Kondrak (2004). The pair HMM (Figure 4.3) has three states that represent basic edit operations: substitution (represented by state “M”), insertion (“Y”), and deletion (“X”). The idea is that when comparing two strings the edit operations M, X and Y are used to transform one string to the other string. In this pair HMM, there are estimated probabilities for emitting symbols while in the states, and for transitions between the states. “M”, the match state has emission probability distribution \( p_{x_i y_j} \) for emitting an aligned pair of symbols \( x_i : y_j \), \( i \) and \( j \) represent indexes in the token set for \( x \) and \( y \) in that order. State X has the emission probability distribution \( q_{x_i} \) for the alignment of a symbol \( x_i \) in the first string against a gap in the second string. State Y has the distribution \( q_{y_j} \) for emitting a symbol \( y_j \) in the second string against a gap in the first string.
The pair HMM model has five transition parameters: $\delta$ represents the probability of going from the substitution state to either the insertion or deletion states; $\varepsilon$ is the probability of staying in the insertion or the deletion state at the next time step; $\lambda$ represents the probability of going from the deletion to the insertion state or from the insertion to the deletion state; $\tau_M$ is the probability of going from the substitution state to the end state; $\tau_{XY}$ is the probability of going from either the deletion or insertion state to the end state. Durbin et al. (1998) chose to tie the probabilities of the start state to those of the substitution state. “The probability of starting in the substitution state is the same as being in the substitution state and staying there, while the probability of starting in the insertion or deletion state equals that of going from the substitution state to the given state” (Mackay, 2004). This set up of initial probabilities for the pair HMM was maintained by Mackay and Kondrak (2005), and later by Wieling et al., (2007).

Two major modifications were made by Mackay and Kondrak (2005) to the model developed by Durbin et al. (1998): in the first modification, a pair of transitions between the insertion and
deletion states was added. In the other modification, the probability for the transition to the end state $\tau$ was split into two separate values: $\tau_M$ for the match state and $\tau_{XY}$ for the gap states.

### 4.3.1 Proposed modifications to the parameters of the pair HMM

Although most of the assumptions with regard to the pair HMM used by Mackay and Kondrak (2004) and by Wieling et al. (2007) are maintained, there are a few obvious modifications that are necessary in applying the pair HMM to transliteration related tasks. Firstly, Mackay and Kondrak (2004) maintain the assumption used by Durbin et al. (1998) concerning the parameter $\delta$, that is, the probability of transiting from the substitution state to each of the gap states is the same. Similarly, the probability of staying in each of the gap states is the same. These assumptions are reasonable for the case where the alphabets of both the source and target languages are identical (Mackay, 2004) for example Dutch and English. If the source and target language alphabets are not identical for example in the case of English and Russian, then the parameters associated with emissions, transitions to and from the two gap states should be distinct. Consequently, each gap state will be associated with different transition parameters and emission parameters for tokens from one language alphabet that are different from the transition parameters and emission parameters of tokens from the other language alphabet in the other gap state. Thus, the transition parameter from the match state to one gap state associated with a one language alphabet $V_1$ should be different from the transition parameter from the match state to the other gap state associated with language alphabet $V_2$. Likewise, the transition parameter of staying in one gap state should be different from the transition parameter of staying in the other gap state, and the transition parameters between the gap states should be different. As a result, the pair HMM that is proposed for similarity estimation between candidate transliterations should have parameters illustrated in figure 4.4.
Figure 4.4 proposed parameters for the pair HMM

With regard to the proposed modifications, one modification associated with having different emission parameters in each of the gap states has been implemented. The assumptions concerning the transition parameters have left as used in previous work (Mackay and Kondrak, 2004; Wieling et al., 2007) with the reason that similarity is mostly dependent on the emission parameters.

4.4 Application of pair HMM to bilingual entity name identification

The pair HMM is applied in such a way that it computes a similarity score for input comprising bilingual entity names from languages with either the same writing system or different writing systems. When computing the string similarity score, the pair HMM uses values of initial, transition, and emission parameters. Table 4.1 is an example illustrating the alignment between two entity names: “peter” obtained from English data and “пёрг” obtained from Russian data. The parameters needed to compute the similarity score for the alignment in table 4.1 are illustrated in expression 4.3.
Table 4.1 operation using the pair HMM

\[
similarity\ score = P(M_0) \times P(e : \pi) \times P(M \rightarrow M) \times (e : \ddot{e}) \times P(M \rightarrow M) \times P(t : T) \\
\times P(M \rightarrow X) \times P(e : \_\_ \_\_) \times P(X \rightarrow M) \times P(r : p) \times P(M \rightarrow END)
\] (4.3)

As shown in the expression (4.3), we need: initial state probabilities, transition probabilities and emission probabilities for computing the similarity scores for a pair of bilingual entity names. Parameter estimation for pair HMMs is next discussed.

### 4.4.1 Parameter Estimation in pair HMMs

Arribas-Gil et al. (2005) reviewed different parameter estimation approaches for pair HMMs including: numerical maximization approaches, Expectation Maximization (EM) algorithm and its variants (Stochastic EM, Stochastic Approximation EM). According to Arribas et al. (2005), “pair HMMs and the standard HMMs are not very different and classical algorithms such as the forward-backward or Viterbi algorithms are still valid and efficient in the pair HMM context”. Arribas-Gil et al. (2005) prove that Maximum Likelihood estimators are more efficient and produce better estimations for pair HMMs on a simulation of estimating evolutionary parameters. In the pair HMM software tool kit adapted for transliteration, the Baum Welch (BW) algorithm (Baum et al., 1972) has been implemented for estimating parameters used by different pair HMMs and is also adopted for the transliteration tasks. The BW algorithm falls under the EM class of algorithms that all work by guessing initial parameter values, then estimating the likelihood of the data under the current probabilities. These likelihoods can then be used to re-estimate the parameters. The re-estimation is done iteratively until a local maximum or a stopping criterion is reached.

### 4.4.2 Baum-Welch Algorithm for the pair HMM

The general intuition behind the BW algorithm for HMMs is as follows (Durand and Hoberman, 2006): we start by choosing some initial values of the starting parameters \( \{ \pi_i \} \), transition parameters \( \{ p_{ij} \} \), and emission parameters \( \{ e_i(v_k) \} \) for the HMM according to some
Probable state paths are then determined using Forward and Backward algorithms (Rabiner, 1989); Counts can then be determined for transiting from state $s_i$ to state $s_j$ ($A_{ij}$) for all states in the model, and for emitting a symbol $v$ in state $s_i$ ($E_i(v)$) for all symbols for a given alphabet in each state; The counts can then be used in determining new parameters $\mathcal{P} = (\{\pi_i\}, \{\bar{P}_{ij}\}, \{\bar{e}_i(v_k)\})$. Baum et al. (1970) proved that with the new estimates, the likelihood of the data does not decrease, that is $P(O \mid \mathcal{P}) \geq P(O \mid \mu)$. Equality will occur if the initial model $\mu$ represents a critical point such as a local maximum that is any point where all partial derivatives are zero (or some partial derivatives do not exist). Through repeated iteration, we get increasingly better models with respect to our training data until we reach some stopping criterion.

For a pair HMM, training data comprises of a set of pairs of observations sequences, $O_1^1 : O_2^1$, $O_1^2 : O_2^2$, $O_1^3 : O_2^3$, ... where $O_1^i : O_2^i$ is such that we have two observations $O_1$ and $O_2$ from state $s_i$ at a given time. We need to search through a 2-dimensional space of possible alignments over different observation pairs. Following the intuition above, we start by choosing initial arbitrary values for the parameters of the pair HMM. At each pair $O_x^d : O_y^d$ in the observation sequence, we calculate the most probable path using the forward and backward variables. The forward and backward variables are calculated using the forward and backward algorithms respectively. With reference to figure 4.3, the forward and backward algorithms for the pair HMM are as shown in Table 4.2 and Table 4.3 respectively.

1. Initialization
   
   $f^M(0, 0) = 1 - 2\delta - \tau_M$, $f^X(0, 0) = f^Y(0, 0) = \delta$
   
   All $f^*(i, -1), f^*(-1, j) = 0$.

2. Induction: for $0 \leq i \leq n$, $0 \leq j \leq m$, except $(0, 0)$
   
   $f^M(i, j) = p_{x,y} \left[ (1 - 2\delta - \tau_M) f^M(i-1, j-1) + (1 - \epsilon - \lambda - \tau_{xy})(f^X(i-1, j-1) + f^Y(i-1, j-1)) \right]$,  
   
   $f^X(i, j) = q_y \left[ \delta f^M(i-1, j) + \epsilon f^X(i-1, j) + \lambda f^Y(i-1, j) \right]$,  
   
   $f^Y(i, j) = q_x \left[ \delta f^M(i, j-1) + \lambda f^X(i, j-1) + \epsilon f^Y(i, j-1) \right]$

3. Termination
   
   $P(O \mid \mu) = \tau_M f^M(n, m) + \tau_{xy}(f^X(n, m) + f^Y(n, m))$.

Table 4.2 Forward algorithm for pair HMM (Adapted from Mackay, 2004)
1. **Initialization**
\[
b^M(n,m) = \tau_M, b^X(n,m) = b^Y(n,m) = \tau_{XY}.
\]
All \(b^*(i,m+1), b^*(n+1,j) = 0\).

2. **Induction**
\[
b^M(i,j) = (1 - 2\delta - \tau_M)p_{s_{i+1}s_{j+1}}b^M(i+1,j+1) + \delta(q_{s_{i+1}}b^X(i+1,j) + q_{s_{j+1}}b^Y(i,j+1)),
\]
\[
b^X(i,j) = (1 - \epsilon - \lambda - \tau_{XY})p_{s_{i+1}s_{j+1}}b^M(i+1,j+1) + \epsilon q_{s_{i+1}}b^X(i+1,j) + \lambda q_{s_{j+1}}b^Y(i,j+1),
\]
\[
b^Y(i,j) = (1 - \epsilon - \lambda - \tau_{XY})p_{s_{i+1}s_{j+1}}b^M(i+1,j+1) + \lambda q_{s_{i+1}}b^X(i+1,j) + \epsilon q_{s_{j+1}}b^Y(i,j+1),
\]

3. **Termination**
\[
P(O\mid \mu) = (1 - 2\delta - \tau_M)b^M(0,0) + \delta(b^X(0,0) + b^Y(0,0)).
\]

Table 4.3 Backward algorithm for pair HMM (Adapted from Mackay, 2004)

In Tables 4.2 and 4.3, \(\star\) represents an action performed for all the states: M, X, and Y. \(p_{s_{i}s_{j}}\) is the probability of matching a character at position \(i\) in the observation stream \(x\) (i.e. string \(x\) of length \(n\)) with a character at position \(j\) in the string \(y\) of length \(m\) while in the match state M. Likewise, \(q_{s_{i}}\) is the probability of matching a character at position \(i\) in string \(x\) with a gap in string \(y\) while in the gap state X, and \(q_{s_{j}}\) is the probability of matching a gap in string \(x\) with a character at position \(j\) in string \(y\) while in the gap state Y.

The maximum likelihood estimators for the transition and emission probabilities can be given by:
\[
\bar{a}_{kl} = \frac{a_{kl}}{\sum_{l'}a_{kl'}}\quad \text{and} \quad \bar{e}_k(v^{(x_i,y_j)}) = \frac{e_k(v^{(x_i,y_j)})}{\sum_{(x_i,y_j)}e_k(v^{(x_i,y_j)})}\quad \text{respectively.}
\]
\(x_i\) is an element from an alphabet \(V_1\) associated with one of the languages and \(y_j\) is an element from an alphabet \(V_2\) associated with the other language. \(a_{kl}\) represents the number of transitions from a state \(k\) to \(l\). \(e_k(v^{(x_i,y_j)})\) is the number of emissions of the pair \((x_i,y_j)\) from state \(k\). For the gap states, emissions comprise aligning a symbol against a gap, i.e. \(\{x_i;\_\}\), \(\{\_y_j\}\).

To derive the expressions necessary for computing the pair HMM parameters we first look at the case of the single observation sequence HMMs, which are simply referred to as HMMs here.

For HMMs (Rabiner, 1989), the probability of transiting from a state \(s_i\) to a state \(s_j\) at time \(t\) is usually specified by a variable \(\xi_t(i,j)\) and is given as
$$\xi_t(i, j) = P(q_t = s_i, q_{t+1} = s_j | O^d, \mu) = \frac{P(q_t = s_i, q_{t+1} = s_j, O^d | \mu)}{P(O^d | \mu)} \quad (4.4)$$

where $O^d$ is an observation sequence and $\mu$ is a given HMM.

Through expansion and simplification using forward $(\alpha_t(t))$ and backward $(\beta_t(t))$ variables for HMMs, it can be shown that (Mackay, 2004):

$$\xi_t(i, j) = \frac{\alpha_t(i) p_y e_j(o_{t+1}^d) \beta_{t+1}(j)}{P(O | \mu)} \quad (4.5)$$

The probability of being in state $i$ at time $t$ is also usually specified by a variable $\gamma_t(i)$:

$$\gamma_t(i) = P(q_t = s_i | O^d, \mu) = \frac{P(q_t = s_i, O^d | \mu)}{P(O^d | \mu)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)} \quad (4.6)$$

It can be seen that

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j) = \sum_{j=1}^{N} \frac{\alpha_t(i) p_y e_j(o_{t+1}^d) \beta_{t+1}(j)}{P(O | \mu)} = \frac{1}{P(O | \mu)} \alpha_t(i) \left[ \sum_{j=1}^{N} p_y e_j(o_{t+1}^d) \beta_{t+1}(j) \right]$$

$$= \frac{1}{P(O | \mu)} \alpha_t(i) \beta_t(i) \quad (4.7)$$

If we sum over the time index for the two variables: $\xi_t(i, j)$ and $\gamma_t(i)$, we get expectations (or counts) that can be used in re-estimating the parameters of a HMM with the following reasonable re-estimation formulas:

$$\bar{\pi}_t = \text{expected number of times in state } i \text{ at time } t = 1 \quad \gamma_t(i) \quad (4.8)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{1}{P(O^d)} \sum_{t=1}^{T} \alpha_t(i) p_y e_j(o_{t+1}^d) \beta_{t+1}(j) \quad (4.9)$$

$$\bar{c}_t(v) = \frac{\text{expected number of times in state } i \text{ observing symbol } v}{\text{expected number of times in state } i}$$

$$= \frac{\sum_{t=v, 1 \leq t \leq T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} = \frac{1}{P(O^d)} \sum_{v \in O^d} \alpha_t(i) \beta_t(i) \quad (4.10)$$
In the case of pair HMMs, we have to sum over each pair position, and over all possible sequences. If for a pair HMM, \( h \) represents the index of the pair we are using and the forward and backward variables are \( f \) and \( b \) as above; we obtain the following expressions for the transition and emission estimations in the different states:

For transition ending in the substitution state we have

\[
\begin{align*}
\bar{a}_{kl} &= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(x_{i+1}^h, y_{j+1}^h) b_{(i+1,j+1)}^b(l) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(x_{i+1}^h, y_{j+1}^h) b_{(i+1,j+1)}^b(l) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(x_{i+1}^h, y_{j+1}^h) b_{(i+1,j+1)}^b(l)
\end{align*}
\] (4.11)

For an emission in the substitution state

\[
\begin{align*}
\bar{c}_k(v^{(i,j)}) &= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j) b_{(i, j)}^b(k) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j) b_{(i, j)}^b(k) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j) b_{(i, j)}^b(k)
\end{align*}
\] (4.12)

The equations for the gap states will have slightly different forms (Mackay, 2004). For example, in the gap state \( Y \), we only need to match the symbol \( y_j \), since \( y_j \) is emitted against a gap. In the estimation for the transition probability, when we end in a gap state, we only change the index for one of the pairs and we use the emission probability for a symbol from one string against a gap. The remaining estimations should then be:

For the gap state \( X \):

\[
\begin{align*}
\bar{A}_{kl} &= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(x_{i+1}^h) b_{(i+1,j+1)}^b(l) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(x_{i+1}^h) b_{(i+1,j+1)}^b(l) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(x_{i+1}^h) b_{(i+1,j+1)}^b(l)
\end{align*}
\] (4.13)

\[
\begin{align*}
\bar{E}_k(O^{xy}) &= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j) b_{(i, j)}^b(k) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j) b_{(i, j)}^b(k) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j)
\end{align*}
\] (4.14)

For the gap state \( Y \):

\[
\begin{align*}
\bar{A}_{kl} &= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(y_{j+1}^h) b_{(i+1,j+1)}^b(l) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(y_{j+1}^h) b_{(i+1,j+1)}^b(l) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(k) p_{kl} e_i(y_{j+1}^h) b_{(i+1,j+1)}^b(l)
\end{align*}
\] (4.15)

\[
\begin{align*}
\bar{E}_k(O^{xy}) &= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j) b_{(i, j)}^b(k) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j) b_{(i, j)}^b(k) \\
&= \sum_h \frac{1}{P(O \mid \mu)} \sum_i \sum_j f_{(i,j)}^h(i, j)
\end{align*}
\] (4.16)

With these equations, we can specify an algorithm that is able to learn all of the parameters of the pair HMM. All we need now is to provide the algorithm with training data representative of the similarity that we want to model.
4.4.3 pair-HMM Training Software

For estimating the parameters of a pair HMM for application to transliteration, the pair HMM software used by Wieling (2007) has been adapted. Wieling’s (2007) pair HMM training software was modified to use two alphabets, which are denoted here as follows: $V_i = \{v_{ij}\}$ for $i = 1, \ldots, m$ is the alphabet of symbols in one writing system; and $V_2 = \{v_{2j}\}$ for $j = 1, \ldots, n$ is the alphabet of symbols in the other writing system. Each alphabet that is used by the pair HMM is generated automatically from the data set for the corresponding language. The other modification made to the pair HMM training software is concerned with how the data used for training is arranged. In Wieling’s (2007) pair HMM training software, the requirement for data representation is that matching names in different dialects that are used for training are combined and located in separate files. This was more efficient in Wieling’s (2007) case because of the relatively larger number of dialects (which correspond to languages here) that were being analyzed relative to the number of names considered in each dialect. However, the requirement should be reversed for the pair HMM software adapted for transliteration to reduce on the number of opening and closing processes relative the number of languages used (in this case only two). The pair HMM software was therefore modified to use only two files and each file holds all the names from one language with matching named entities located at the same position in the two files.

4.5 Experiments

Different experiments have been carried out with regard to using the pair HMM for identifying bilingual named entities. In one set of experiments, we determine the accuracy of using the forward pair HMM algorithm against the Viterbi pair HMM algorithm in estimating the similarity between bilingual named entities for both cases of languages that use the same writing system and languages that use different writing systems. In the second set of experiments, the pair HMM is applied in automatically learning parameters for identifying transliteration pairs from Wikipedia cross language articles. In particular, for the second set of experiments, the forward pair HMM algorithm is evaluated against the forward log-odds algorithm for identifying English-Russian transliteration pairs from cross language English-Russian article pages.
4.5.1 Forward pair HMM algorithm vs. Viterbi pair HMM algorithm

After estimating the parameters that the pair HMM can use, different algorithms can be used in computing the scores of the observation sequence for a given task. In this work two algorithms are used: the forward algorithm (Table 3.2 and 3.3) that takes into account all possible alignments to calculate the score associated with estimating the similarity of two words; and the Viterbi algorithm that considers only the best alignment to compute the similarity score. The Forward algorithm for the pair HMM has been introduced in section 4.4.3. The Viterbi algorithm is introduced as follows.

Like in the cases of Forward and Backward algorithms, we define a variable
\[ \delta_i(t) = \max_{q_0,...,q_i} P(q_1,...,q_{i-1},a_1^d...a_i^d,q_i = s_i | \mu) \]
that represents the maximum probability of all sequences ending at a state \( i \) at time \( t \) given the model. This variable can be calculated recursively to get the single best path that accounts for all of the observations (Mackay, 2004).

To retrieve the state sequences, we need to keep track of the path through the model using an array \( \psi_i(j) \). The procedure needed to calculate for the two variables associated with the Viterbi algorithm can be found in (Mackay, 2004).

1. Initialization
   \[
   \nu^M(0,0) = 1 - 2\delta - \tau, \quad \nu^X(0,0) = \nu^Y(0,0) = \delta \\
   \text{All } \nu^*(i,-1), \nu^*(-1,j) = 0.
   \]

2. Induction: for \( 0 \leq i \leq n, \ 0 \leq j \leq m \), except \( (0,0) \)
   \[
   \nu^M(i,j) = p_{xy} \max \left\{ (1 - 2\delta - \tau)\nu^M(i-1,j-1), \ (1 - \epsilon - \lambda - \tau_{xy})\nu^X(i-1,j-1), \ (1 - \epsilon - \lambda - \tau_{xy})\nu^Y(i-1,j-1) \right\},
   \nu^X(i,j) = q_{xy} \max \left\{ \delta \nu^M(i-1,j), \epsilon \nu^X(i-1,j) \right\},
   \nu^Y(i,j) = q_{xy} \max \left\{ \delta \nu^M(i-1,j), \epsilon \nu^Y(i-1,j) \right\},
   \nu^M(i,j) = p_{xy} \max \left\{ \delta \nu^M(i-1,j), \epsilon \nu^X(i-1,j), \lambda \nu^Y(i-1,j) \right\},
   \]

3. Termination
   \[
P(X^*) = \max(\tau_{xy}\nu^M(n,m), \tau_{xy}\nu^X(n,m), f^Y(n,m))
   \]

Table 4.4 Viterbi algorithm for pair HMM (Adapted from Mackay, 2004)
With reference to figure 4.3, the pseudo code for the Viterbi algorithm adapted from Mackay (2004) is as shown in table 4.4. Again, the symbol * is used to represent an action performed for all states M, X, and Y. For all the algorithms used in the pair HMM, the input is a pair of names (the observation sequence), and the output should be a score for the pair of names determined at the termination of the algorithm.

### 4.5.1.1 Training data and Training time

The training data used for this set of experiments consisted of matching pairs of names extracted from the GeoNames\(^3\) and Wikipedia data dumps. The names comprise mainly of location names and a small percentage of person names. Entity names with spaces in them were not considered; each pair extracted has only one entity name from each language without a space. In total, 850 English-French pairs of names were extracted and 5902 English-Russian pairs of names were extracted. No particular ratio was followed for dividing the data sets obtained into training and testing or evaluation sets; instead, as many names as possible were reserved for training and the rest for testing. For the English-French dataset, 600 entity name pairs were used for training. From the English-Russian dataset, 4500 pairs were used for training. Using the English-French training set, it took 282 iterations in approximately five minutes for the training algorithm to converge while when using the English-Russian data sets, it took 848 iterations for the algorithm to converge for an initial setting of uniform probabilities.

### 4.5.1.2 Evaluation

Two measures have been used for evaluating the accuracy of the forward and Viterbi algorithms: Average Reciprocal Rank (ARR) and Cross Entropy (CE).

#### Average Reciprocal Rank Results

To determine the accuracy of a system in identifying bilingual named entities, two related measures proposed by Voorhees and Tice (2000) can be used: Average Rank (AR) and the Average Reciprocal Rank (ARR) (or Mean Reciprocal Rank (MRR)):

\[
AR = \frac{1}{N} \sum_{i=1}^{N} R(i)
\]

\[
ARR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{R(i)}
\]

---

\(^3\) http://download.geonames.org/export/dump/
\[ ARR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{R(i)} \]  

where \( N \) is the total number of unique entity names in the testing dataset, and \( R(i) \) is the rank of the correct matching entity name for a particular source entity name in the set of answers associated with the \( i^{th} \) source entity name in the testing data. The value of ARR ranges from 0 and 1. If for each source entity name, there is only one correct matching target entity name that is considered, then the closer ARR is to 1, the more accurate is the model is.

For the English-French dataset, the logarithmic versions of the forward and backward algorithms were tested. For the English-Russian data set, the Viterbi and Forward algorithms in their basic form and the log versions of the two algorithms were tested. The log versions of the algorithms are used to deal with any errors that could arise due to assigning very low probabilities to input string pairs. Table 4.5 shows sample results that were obtained from the English-French testing. In Table 4.5, the first column shows the English target name against which French data is to be compared. The second column shows the actual matching French name and the fourth column shows the ranking for the match obtained from the pair HMM similarity scores. For example for the English name “Klausen”, the actual matching French name is “Chiusa” which was ranked fifth against other French names in the test data.

The ARR results are shown in Table 4.6 and Table 4.7 for the English-French and English-Russian test data respectively.

<table>
<thead>
<tr>
<th>English target name</th>
<th>French source name</th>
<th>forward log score</th>
<th>rank</th>
<th>reciprocal rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>chiomonte</td>
<td>chaumont</td>
<td>-18.2444</td>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td>cumae</td>
<td>Cumes</td>
<td>-12.2731</td>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td>empúries</td>
<td>empuries</td>
<td>-24.6083</td>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td>gaeta</td>
<td>Gaëte</td>
<td>-10.0030</td>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td>girona</td>
<td>Gérone</td>
<td>-12.2533</td>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td>kraków</td>
<td>cracovie</td>
<td>-25.6732</td>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td>lüsen</td>
<td>Luson</td>
<td>-13.2366</td>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td>klausen</td>
<td>Chiusa</td>
<td>-22.3873</td>
<td>5</td>
<td>0.2000</td>
</tr>
<tr>
<td>sterzing</td>
<td>vipiteno</td>
<td>-43.8903</td>
<td>( \approx 100 )</td>
<td>0.0100</td>
</tr>
</tbody>
</table>

\[ ARR \approx 0.8000 \]

Table 4.5 Sample results after using forward-log algorithm
### Table 4.6 ARR results for English-French test data

<table>
<thead>
<tr>
<th>Pair HMM Algorithm</th>
<th>ARR (for N = 164)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi-log</td>
<td>0.8099</td>
</tr>
<tr>
<td>Forward-log</td>
<td>0.8081</td>
</tr>
</tbody>
</table>

The ARR results for the log versions for both the Viterbi and Forward algorithms show that the Viterbi algorithm consistently performs slightly better than the Forward algorithm. However, for the English-Russian data set, the ARR results show that the basic Forward algorithm performs slightly better than the basic Viterbi algorithm.

### Table 4.7 ARR results for English-Russian test data

<table>
<thead>
<tr>
<th>pair HMM Algorithm</th>
<th>ARR (for N = 966)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>0.8536</td>
</tr>
<tr>
<td>Forward</td>
<td>0.8545</td>
</tr>
<tr>
<td>Viterbi-log</td>
<td>0.8359</td>
</tr>
<tr>
<td>Forward-log</td>
<td>0.8355</td>
</tr>
</tbody>
</table>

**Cross Entropy for transliteration model comparison**

In this section we introduce the use of cross entropy as a measure for evaluating accuracy of two or more models for estimating the similarity between bilingual named entities. But first, we look at entropy, from which cross entropy is derived. In information theory, entropy quantifies the uncertainty involved when encountering a random variable $X$. The random variables range over whatever we are predicting (for example entity names, characters, etc.) and has a particular probability function, call it $p(x)$. If we have a set of events whose probabilities of occurrence are $p(x_1), p(x_2), \ldots, p(x_n)$. There can be a need to measure how much uncertainty is associated with the events. Such a measure, say $H(x)$ (Shannon, 1948) should have the following properties: (1) $H(x)$ should be continuous in the $p(x_i)$. (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 $H(x)$. The only $H(x)$ satisfying the three assumptions is of the form (Shannon, 1948):

$$H(x) = -K \sum_{i=1}^{n} p(x_i) \log p(x_i)$$  \hspace{1cm} (4.19)
where $K$ is a positive constant. $H(x)$ is referred to as Entropy of the probability distribution over the events. The choice of the logarithmic base will correspond to the choice of the unit for measuring information. For a sequence of observations $S = \{s_1, s_2, ..., s_n\}$, entropy can be computed. For example, the entropy for observing characters in the string can be written as:

$$H(s_1, s_2, ..., s_n) = - \sum_{S_i \in A} p(S_i^n) \log p(S_i^n)$$

(4.20)

where $A$ is an alphabet of characters

A variation of Entropy that enables us to compare two or more models what is referred to as Cross Entropy. Cross Entropy allows us to use some model $m$, which is a model of $p$ (that is, an approximation to $p$). The cross entropy of two probability distributions $p$ and $m$ for a random variable $X$ is written as:

$$H(p, m) = - \sum_i p(x_i) \log m(x_i)$$

(4.21)

Cross Entropy, is however, not a symmetric function, that is $H(p, m) \neq H(m, p)$. The cross entropy $(p, m)$ is also an upper bound on the true entropy $H(p)$. For any model $m$: $H(p) \leq H(p, m)$. If $p = m$, the 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 more accurate the model $m$ (or the better $m$ is an approximation of $p$). Cross Entropy can therefore be used to compare approximate models. Between two models $m_1$ and $m_2$, the more accurate model will be the one with the lower cross entropy.

For the pair HMM, two algorithms have been used for similarity estimation: the forward and Viterbi algorithms. The task is to identify which algorithm best estimates the similarity between the two strings. The cross entropy of the pair HMM on the probability of observing a pair of sequences is given as:

$$H(p, m) = - \sum_{A_1, A_2} p(s_1 : t_1, ..., s_T : t_T) \log m(s_1 : t_1, ..., s_T : t_T)$$

(4.22)

We draw the pairs of sequences according to the probability distribution $p$, but sum the log of their probabilities according to $m$. 33
One major limitation in computing the cross entropy is the lack of the target distribution (that is \( p(s_1 : t_1, ..., s_r : t_r) \)). If a corpus is available with a true representation of the variables, it can be exploited to enable comparison of two or more models using cross entropy. The notion of Corpus Cross Entropy (CCE) (log probability) is used. 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 as:

\[
H_C(m) = -(1/n) \sum_i \log m(c_i))
\]  

(4.23)

where summation is done over tokens in the corpus.

It can be proved that as \( n \) tends to infinity, the CCE becomes the Cross Entropy for the true distribution that generated the corpus. 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 Maximum Likelihood estimate goes to the true probability distribution as the size of the corpus goes to infinity. It is not exactly correct to use the result for cross entropy in NLP application because the above assumption is clearly wrong for languages (Manning and Schutze, 2001). Nonetheless, for a given corpus, we can assume that a language is near enough to unchanging. This will be an acceptable approximation to truth (Askari, 2006). A major advantage that results from utilizing corpus cross entropy is that there is no need for a test corpus as is the case for Average Reciprocal Rank (ARR).

For the pair HMM case, the corpus consists of pairs of entity names. Each pair of entity names \( (s^i; t^i) \) is a token \( c_i \) in the corpus. The log probability of a pair HMM algorithm \( m \) on the corpus can then be written as:

\[
H_c(m) = -(1/n) \sum_{i=1}^n \log(m(s^i, t^i))
\]  

(4.24)

where \( m(s^i, t^i) \) is the estimated probability according to a model \( m \) for the pair of names \( (s^i; t^i) \).

It is also possible to consider character alignments to constitute tokens in the corpus. In that sense, \( (s^i; t^i) \) will be the \( i^{th} \) character alignment in the corpus. Estimating character alignments, however, can only be possible if the corpus comprises aligned characters, moreover manually corrected. In either of the cases above (i.e. whether character alignments or pairs of entity names)
it is important to see whether there is any chance of the log probability converging with increase in corpus size. Figure 4.6 shows the variation of the corpus cross entropy with corpus size for the pair HMM Forward and Viterbi algorithms using a pairs of entity names as tokens. Table 4.8 shows the corpus cross entropy for the two algorithms with the corpus size at 1000. The Corpus Cross Entropy Results suggest that the Forward algorithm is slightly more accurate than the Viterbi algorithm.

### 4.5.1.2 Conclusions based on ARR and CE results

The experimental results show that the accuracy of the pair HMM obtained from using the Forward and Viterbi algorithms is within the range of the accuracy of some of the methods used in previous work such as Weighted Edit Distance Algorithm (WEDA) and statistical techniques in Statistical Transliteration Model (STM) by Lee et al. (2003). The evaluation carried out so far regarding the pair HMM is not sufficient to give critical information regarding the performance of the pair HMM. For example, it should be interesting to determine how sensitive the results are
with regard to the test set size. Nevertheless, the accuracy results obtained are encouraging for extending research associated with the pair HMM. With the ability for the pair HMM to consider different alphabets, it is possible to add a transliteration module that utilizes the parameters estimated using the pair HMM. Later on in the research, I intend to do a more substantial evaluation of the pair HMM including different application oriented evaluations.

4.5.2 Identification of bilingual Entity Names from Wikipedia: Forward algorithm vs. Forward Log Odds algorithm

In this section, the pair HMM is proposed for application to estimating similarity while extracting transliteration pairs from Wikipedia. A requirement for estimating the similarity between candidate transliterations is a model that simplifies the analysis of input strings that are represented using different writing systems. Most of the methods that have been developed for similarity estimation between named entities in a bilingual named entity identification or extraction task require using the same alphabet or the same input representation for the two languages involved. Such methods include: LCSR (Melamed, 1995), Dice Coefficient (Smadja et al., 1996), ALINE (Kondrak, 2003), PREFIX, LCSF (Kondrak and Dorr, 2004), SVM (Bergsma and Kondrak, 2005). Most of these methods can not be applied on input that is represented using completely different writing systems. The pair HMM introduced in section 4.3 enables an almost direct estimation of similarity between two strings represented using different writing systems. Such a model becomes much more valuable in a bilingual named entity extraction task when the two languages use different writing systems. Two pair HMM algorithms (Forward algorithm and Forward log-odds algorithm) are evaluated for extracting transliteration pairs from English and Russian Wikipedias.

4.5.2.1 Approach for Extracting transliteration pairs from Wikipedia

Figure 4.7 illustrates the approach used for transliteration pair extraction from Wikipedia. First, cross language links are used to extract matching named entities that are reserved for training the pair-HMM. The search pattern here is restricted to matching a person name from the source language Wikipedia infoboxes; the matching transliteration is extracted using a cross language link that exists on the same source language Wikipedia article page. After extracting training data, corresponding cross language Wikipedia article pages are used to extract candidate named entities. Cross language Wikipedia article pages with a sufficient word size are identified and
candidate named entities are extracted from each of the article pages. While in previous work on extracting from Wikipedia, the link structure is mostly used for extracting potential candidate named entities, here, the unlinked text is also considered for extracting candidate named entities. A pair HMM algorithm is then applied to estimate similarity between candidate named entities relative to other pairs of named entities. The highest ranking pairs are considered as potential matching transliteration pairs.

4.5.2.2 pair HMM Forward Log Odds Algorithm

The pair HMM Forward log odds algorithm is an extension to the forward algorithm that was introduced in section 4.3. Forward log odds algorithm estimates the score for a pair of transliterations by dividing the similarity score estimated using the pair HMM forward algorithm by the similarity score obtained by first generating the source language transliteration and target language transliteration one after the other, through a random model (figure 4.8). The random model has only one parameter as shown in figure 4.8.
4.5.2.3 Experimental Setup

Training Data
Training data is extracted from English Wikipedia database dump downloaded on 8th December 2008. The extraction approach involves defining regular patterns to take advantage of the structured information existing in Wikipedia infoboxes. The search is restricted to only person names. A total of 6593 English-Russian transliteration pairs are extracted for training. Alphabets used by the pair HMM are obtained from the extracted collection of person names for each language. For English, 95 characters are used and for Russian, 72 characters are used.

Testing Data
Five online English wikipedia articles with the titles “Abraham Lincoln”, “Dennis Bergkamp”, “Guus Hiddink”, “Marat Safin”, “Yoweri Museveni”, and the correspond (cross language) Russian Wikipedia articles were identified. Transliteration candidates were then extracted from the article pairs and prepared as input to the pair HMM for estimating similarity. In total, 2101 English candidate names, and 1310 Russian candidate names were used. Matching transliterations from each of the article pairs were hand picked to form a gold standard set with 200 transliteration pairs.

Evaluation
The measure used for evaluating the accuracy of the pair HMM forward and forward log odds algorithms is precision ($p$).
\[ p = \frac{\text{number of correct matching names returned for names in the test set}}{\text{total number of names (N) in the test set}} \]

Table 4.9 illustrates the precision values at the first rank for the pair HMM forward algorithm.

<table>
<thead>
<tr>
<th>English</th>
<th>Correct Russian match</th>
<th>Precision at 1\textsuperscript{st} rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arkansas</td>
<td>Арканзас</td>
<td>1</td>
</tr>
<tr>
<td>jefferson davis</td>
<td>джефферсона дэвиса</td>
<td>0(2)</td>
</tr>
<tr>
<td>william seaward</td>
<td>уильям сьюард</td>
<td>1</td>
</tr>
<tr>
<td>lincoln memorial</td>
<td>мемориал линкольна</td>
<td>0(&gt;100)</td>
</tr>
<tr>
<td>Maryland</td>
<td>Мэриленд</td>
<td>1</td>
</tr>
<tr>
<td>Louisiana</td>
<td>Луизиана</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.9. Sample results for precision at 1\textsuperscript{st} Rank

Table 4.10 shows the precision values for the forward and the forward log odds algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( p \ (1\textsuperscript{st} \text{ Rank}) )</th>
<th>( p \ (2\textsuperscript{nd} \text{ Rank}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward</td>
<td>0.755</td>
<td>0.875</td>
</tr>
<tr>
<td>Forward log odds</td>
<td>0.760</td>
<td>0.900</td>
</tr>
</tbody>
</table>

Table 4.10. Precision values for the forward and forward log-odds algorithm for \( N = 200 \).

Precision values suggest that the forward log-odds algorithm is better than the basic forward algorithm when applied to identifying transliteration pairs.

4.5.2.4 Conclusion

Precision values show a promising application of the pair HMM in extracting transliteration pairs. However, the extraction system based on the pair HMM is not able to match named entities that are ordered differently across different languages for example “lincoln memorial” and “мемориал линкольна” as shown in Table 4.10. For such a case, a more sophisticated translation model will be needed. There are some applications that do benefit from this specific experiment. One recognizable application is the identification of transliteration pairs for extending the link structure in Wikipedia; as an example, a third of the transliteration pairs extracted from the cross language article pages on “Yoweri Museveni” were not linked.
4.6 Future work with regard to pair HMMs for recognition of bilingual entity names

Further work related to the pair HMM for estimating the similarity between bilingual entity names can take different directions.

First of all it is important, that the pair HMM be evaluated against other models for the entity name recognition task. One of the techniques identified against which the pair HMM can be evaluated involves use of Weighted Finite State Transducers (WFSTs) for assigning similarity scores to two strings apart from generating transliterations or translations. The FST techniques are implemented in Carmel (Graehl, 1998) a freely available finite state automata toolkit for manipulating finite state acceptors and transducers. Other methods include discriminative methods for example those used by Bergsma and Kondrak (2007) for string similarity estimation.
Chapter 5

Transliteration Generation using pair HMM with WFSTs

5.1 Introduction
This chapter introduces various Weighted Finite State Transducer (WFST) models for generating transliterations. One set of WFST models utilizes parameters learned for a pair HMM in a transliteration system that was developed to participate in a transliteration shared task (NEWS2009, 2009). Another set of WFST models takes on the structure of the pair HMM, but do not use the parameters learned for a pair HMM; these WFSTs utilize parameters on which they are trained using a forward-backward EM algorithm. Different WFST models with varying numbers of states are also investigated for transliterations between two languages whose source named entities come from a language using a different writing system. The pair HMM with WFST models are evaluated against a Phrase-Based Statistical Machine Translation (PBSMT) system. The pair HMM has been introduced in section 4.3. We first introduce WFSTs and PBSMT for character-based translation. We then look at the different applications in which pair HMM and WFSTs have been utilized for transliteration generation.

5.2 Weighted Finite State Transducers
A Finite State Transducer (FST) is an automaton that transforms one string into another. It can be seen as a network of states with transitions between them which are labeled with input and output symbols. Starting at some state and walking through the automaton to some end state, the FST can transform an input string (by matching the input labels) to an output string (by printing corresponding output labels). Figure 5.1 shows an example of a FST where each arc is labeled by an input and output strings separated by a colon while the nodes represent states.
Weighted Finite State Transducers (WFSTs) are automata in which each transition in addition to its usual input label is augmented with an output label from a possibly different alphabet, and carries some weight element (Mohri, 2004). Assuming that the weights form proper probability distributions, we can compute the probability of certain transformations and compare them with other competing transformations. In this way the most likely transformation can be chosen according to the model. Although the framework of WFSTs has mostly been applied in representing various models for speech recognition (Mohri et al., 2008), WFSTs have as well been used for generating transliterations (Knight and Graehl, 1998). For the transliteration generation tasks reported here, transformation from a source language name to a target language transliteration is first modeled as a sequence of edit operations on the character level with respect to the structure of a pair HMM. Figure 5.2 shows the structure of the WFST that precisely corresponds to a pair HMM considering the constraints of a pair HMM.

![Finite State Transducer](image)

**Fig. 5.1.** Example of a Finite State Transducer (Adapted from (Mohri, 1997))

**Figure 5.2.** Finite State Transducer corresponding to the pair HMM
In figure 5.2, e is an empty symbol while $x_i$ and $y_j$ are as defined for the pair HMM in figure 4.3. Note that in 5.2, a start state is needed to model pair HMM parameter constraints for starting in any of the three edit states. It is also possible to specify a WFST corresponding to the pair HMM with no start state.

### 5.3 Phrase-based Statistical Machine Translation

Phrase-based Statistical Machine Translation (PBSMT) is the current state of the art in data driven machine translation. It is based on the well-known IBM models trained on large parallel corpora but using bilingual phrase-tables instead of word link probabilities and fertility parameters. In PBSMT, several 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.

PBSMT can be used on character level instead of word level (Mathews, 2007). The entire procedure is directly applicable in the same way as standard word-level PBSMT systems are organized. Instead of aligning words, we have to align characters in parallel data, which are transliterations in this case instead of translated sentences. Phrases now refer to character n-grams instead of word n-grams and language models are also defined over character sequences instead of word sequences.

The advantage of PBSMT is that the extracted phrase tables (character n-grams) will cover a lot of contextual dependencies found in the data. In this way we hope to find better transformations by translating sequences of characters instead of single characters. Furthermore, insertions and deletions are not modeled; this is left for the translation table to change the lengths of translated strings. Another advantage is the inclusion of a specific language model to weight the possible outcomes of the system. For actual translation, standard SMT decoders with monotonic decoding are used.

### 5.4 Transliteration System using parameters learned for a pair HMM

In this section, a system that utilizes parameters learned for a pair HMM in a transliteration generation task is introduced. To generate transliterations using pair HMM parameters, WFST
(Graehl, 1997) techniques are adopted. Transliteration generation is based mainly on the initial orthographic representation and there is no explicit phonetic scheme that is used. Instead transliteration quality is tested for different bigram combinations including all English vowel bigram combinations and n-gram combinations specified for Cyrillic Romanization by the US Board on Geographic names and British Permanent Committee on Geographic names (BGN/PCGN). However, transliteration parameters can still be estimated for a pair HMM when a particular phonetic representation scheme is used. The quality of transliterations generated using pair HMM parameters is evaluated against transliterations generated from training WFSTs and transliterations generated using a PBSMT system.

5.4.1 Machine Transliteration System

The transliteration system comprises of a training component for learning the parameters of a pair HMM and a generation component (Figure 5.3). In the training component, the Baum-Welch expectation maximization (Baum et al., 1970) algorithm is used to learn the parameters of a pair HMM. In the generation component, WFST techniques (Graehl, 1997) are used to model pair HMM parameters and use them for generating transliterations. The pair HMM has already been introduced in section 4.3. Transformation rules are also defined as part of the transliteration system so as to enable modeling of context with the aim of improving transliteration quality.

![Figure 5.3. Transliteration generation system](image)

5.4.2 Transformation Rules

A look into the transliterations generated using pair HMM parameters on English-Russian development data showed consistent mistransliterations mainly due to lack of contextual modeling in the generated transliterations. For example, in all cases where the Russian character
The Russian soft sign \( 'l' \) precedes the Russian soft sign \( 'l' \), the Russian soft sign was missing, resulting into a loss of transliteration accuracy. Two examples corresponding to mistransliterations generated that do not include the Russian soft sign \( 'l' \) are: крефельд instead of крефельд ‘krefeld’, and бильбао instead of бильбао ‘bilbao’. For such cases, simple transformation rules, such as “л→ль” were defined on the output transliterations in a post processing step. 25 transformation rules were manually defined for some of the mistransliterations to determine the effect of modeling context in the generated transliterations.

5.4.3 Experiments

The experiments reported in this section were part of a shared transliteration task. For the shared transliteration task two types of runs were specified: a standard run which involves using a proposed transliteration system on data provided for the shared task; and non-standard runs which involve using other systems or additional data for generating transliterations. The standard run involved using the transliteration system introduced in section 5.2 to 5.4. The data set for the standard run comprised of 5977 English-Russian pairs of names for training, 943 pairs for development, and 1000 pairs for testing. For the non standard runs, English-Russian training and development datasets for the shared transliteration task were merged with English-Russian pairs extracted from the Geonames data dump; 10481 pairs were use as training and development data for the non-standard runs. For a second set of experiments (table 2), a different set of test data (1000 pairs) extracted from the Geonames data dump was used. For the system used in the standard run, the training data was preprocessed to include representation of bigrams associated with Cyrillic Romanization and all English vowel bigram combinations.

5.4.3.1 Evaluation Metrics

For the shared transliteration task (NEWS2009, 2009), 10 best candidate transliterations generated by a given system were required for any given source language word. Since a given source language word may have multiple correct transliterations, all alternatives that are considered correct transliterations are treated equally in the evaluation process. Subsequently, the measures that were used for evaluating systems that participated in the shared transliteration task were developed to consider the case of a source name having multiple correct reference transliterations. Six measures were used for evaluating a system’s transliteration quality. These include (NEWS2009, 2009): Accuracy (ACC), Fuzziness in Top 1 (Mean F Score), Mean
Reciprocal Rank (MRR), Mean Average Precision for reference transliterations, (MAP_R), Mean Average Precision in 10 best candidate transliterations (MAP-10), Mean Average Precision for the system (MAP_sys). The MRR metric was introduced in chapter 4. However the MRR described in this section is an extension to the conventional MRR in that multiple reference transliterations are now considered in the metric.

To describe the evaluation metrics the following notation is used (NEWS2009, 2009):

\[ N : \text{total number of names (source words) in the test set} \]
\[ n_i : \text{number of reference transliterations for } i^{th} \text{ name in the test set } (n_i \geq 1) \]
\[ r_{i,j} : j^{th} \text{ reference transliteration for } i^{th} \text{ name in the test set} \]
\[ c_{i,k} : k^{th} \text{ candidate transliteration (output by the transliteration system) for } i^{th} \text{ name in the test set } (1 \leq k \leq 10). \]
\[ K_i : \text{number of candidate transliterations produced by a transliteration system} \]

**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

\[
ACC = \frac{1}{N} \sum_{i=1}^{N} \left\{ 1 \text{ if } \exists r_{i,j}: r_{i,j} = c_{i,1}; 0 \text{ otherwise} \right\} \quad (5.1)
\]

**Fuzziness in Top-1 (Mean F score)**

The *mean F score* measures how different, on average, the top transliteration candidate 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:

\[
LCS(c, r) = \frac{1}{2}(length(c) + length(r) − ED(c, r)) \quad (5.2)
\]

where \(ED(c, r)\) 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 taken for calculation. If the best matching reference \( r_{i,m} \) is given by \( r_{i,m} = \arg \min_j (ED(c_{i,j}, r_{i,j})) \)

Recall \((R_i)\), Precision \((P_i)\), and F-score \((F_i)\) for the \( i^{th} \) word are calculated as

\[
R_i = \frac{\text{LCS}(c_{i,j}, r_{i,m})}{\text{length}(r_{i,m})} \quad P_i = \frac{\text{LCS}(c_{i,j}, r_{i,m})}{\text{length}(c_{i,j})} \quad F_i = \frac{2R_i \times P_i}{R_i + P_i}
\]

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 at 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.

\[
MRR = \frac{1}{N} \sum_{i=1}^{N} \left\{ \min_j \frac{1}{j} \text{ if } \exists r_{j,k}, c_{i,k} : r_{i,j} = c_{i,k} \text{; } 0 \text{ otherwise} \right\} \quad (5.3)
\]

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.

\[
\text{MAP}_{\text{ref}} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\min(n_i,10)} \left( \sum_{k=1}^{\min(n_i,10)} \text{number of correct candidates for } i^{th} \text{ word in } k\text{-best} \right) \quad (5.4)
\]

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 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 below 10, then MAP_10 can never be equal to 1. Only if the number of reference transliterations for every source word is at least 10, then MAP_10 could possibly be equal to 1.
\[
MAP_{10} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{10} \left( \sum_{k=1}^{10} \text{number of correct candidates for } i\text{-th word; in } k\text{-best} \right)
\]  

(5.5)

MAP\_sys

MAP\_sys measures the precision in 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 variable number of transliterations, based on their confidence in identifying and producing correct transliterations. If all of 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 MAP is 1.

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

(5.6)

5.4.3.2 Results

Table 5.1 shows the results obtained using only the data sets provided for the shared transliteration task. The system used for the standard run is “phmm\_rules” described in sections 5.2 to 5.4. “phmm\_basic” is the system in which pair HMM parameters are used for transliteration generation but there is no representation for bigrams as described for the system used in the standard run. Table 5.2 shows the results obtained when additional data from Geonames data dump was used for training and development. In table 5.2, “WFST\_basic” and “WFST\_rules” are the systems associated with training WFSTs corresponding to phmm\_basic and phmm\_rules systems respectively. Moses\_PSMT is the phrase-based statistical machine translation system. The results in both tables show that the systems using pair HMM parameters perform relatively better than the system trained on WFSTs but not better than Moses. The low transliteration quality in the pair HMM and WFST systems as compared to Moses can be attributed to lack of modeling contextual dependencies unlike the case in Moses.

<table>
<thead>
<tr>
<th>metrics</th>
<th>models</th>
<th>ACC</th>
<th>Mean F Score</th>
<th>MRR</th>
<th>MAP_R</th>
<th>MAP_10</th>
<th>MAP_sys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>phmm_basic</td>
<td>0.293</td>
<td>0.845</td>
<td>0.325</td>
<td>0.293</td>
<td>0.099</td>
<td>0.099</td>
</tr>
<tr>
<td></td>
<td>Moses_PSMT</td>
<td>0.509</td>
<td>0.908</td>
<td>0.619</td>
<td>0.509</td>
<td>0.282</td>
<td>0.282</td>
</tr>
<tr>
<td></td>
<td>phmm_rules</td>
<td>0.354</td>
<td>0.869</td>
<td>0.394</td>
<td>0.354</td>
<td>0.134</td>
<td>0.134</td>
</tr>
</tbody>
</table>

Table 5.1: Results from data sets for shared transliteration task
### Table 5.2: Results from additional Geonames data sets

<table>
<thead>
<tr>
<th>models</th>
<th>metrics</th>
<th>ACC</th>
<th>Mean F Score</th>
<th>MRR</th>
<th>MAP_R</th>
<th>MAP_10</th>
<th>MAP_sys</th>
</tr>
</thead>
<tbody>
<tr>
<td>phmm_basic</td>
<td></td>
<td>0.341</td>
<td>0.776</td>
<td>0.368</td>
<td>0.341</td>
<td>0.111</td>
<td>0.111</td>
</tr>
<tr>
<td>phmm_rules</td>
<td></td>
<td>0.515</td>
<td>0.821</td>
<td>0.571</td>
<td>0.515</td>
<td>0.174</td>
<td>0.174</td>
</tr>
<tr>
<td>WFST_basic</td>
<td></td>
<td>0.321</td>
<td>0.768</td>
<td>0.403</td>
<td>0.321</td>
<td>0.128</td>
<td>0.128</td>
</tr>
<tr>
<td>WFST_rules</td>
<td></td>
<td>0.466</td>
<td>0.808</td>
<td>0.525</td>
<td>0.466</td>
<td>0.175</td>
<td>0.175</td>
</tr>
<tr>
<td>Moses_PSMT</td>
<td></td>
<td>0.612</td>
<td>0.845</td>
<td>0.660</td>
<td>0.612</td>
<td>0.364</td>
<td>0.364</td>
</tr>
</tbody>
</table>

#### 5.4.4 Conclusion

A transliteration system using pair HMM parameters has been presented. Although its performance is better than that of the system based on only WFSTs, its transliteration quality is lower than the PBSMT system. On seeing that the pair HMM generated consistent mistransliterations, manual specification of a few contextual rules resulted in improved transliteration quality. A more general framework, in which contextual issues can be investigated in addition to other factors such as position in the source and target strings, and edit operation memory in transliteration, is that of Dynamic Bayesian Networks (DBNs).

#### 5.5 Translating Transliterations

##### 5.5.1 Introduction

The main motivation for integrating a machine transliteration module in NLP applications is to handle unseen terms in a proper way so that performance of that application is improved. However, a task that is not addressed in the literature at least to our knowledge (Tiedemann and Nabende, 2009) is that of translating transliterated names. With this, we refer to the translation of names that have been translated from a different writing system in a third language. Consider the examples in table 5.3 where Russian names have been transliterated into: English, French, German, and Dutch. As we can see, names are spelled very differently even though all of the four languages use the same writing system (the Roman alphabet). Such spelling variations for the same source name in target languages arise due to language specific differences, for example in the way of encoding pronunciations (Hsu et al., 2007).
Table 5.3 Russian names in four European languages

<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 Khrushchev</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>

The problems addressed by generating transliterations apply here as well. The different spelling variants try to match the underlying phonetic description which is usually not known to a cross-lingual application. A dedicated module for translating (unknown) transliterated entity names is expected to help a system (for example MT) in the same way a transliteration module improves performance across writing systems. In this section WFSTs are evaluated against a PBSMT system for translating Russian names between English and Dutch, and between English and French.

5.5.2 Experiments

5.5.2.1 Data Sets

The data sets used are extracted from an English wikipedia database dump from 2008/07/24. Simple patterns are used to identify Russian names looked at the structured information in Wikipedia info-boxes. We basically look at entries that match the pattern (Russian|Russial|Soviet) in categories such as “citizenship”, “nationality”, and “place of birth”. Translations of these names are taken from links to other languages which are also given for every Wikipedia page. In this way we collect all names potentially from Russian origin and their correspondences in other languages. We save all name pairs for the language pairs we are interested in, performing some extra normalization. For example, the German Wikipedia page referring to “Nikita Khrushchev” includes his middle name (“Nikita Sergejewitsch Chruschtschow”). In order to fix this problem, we use the following heuristics for names with different numbers of space separated elements: firstly, we remove abbreviated middle names (such as “H. W.” in George H. W. Bush); secondly, only the first and the last element are considered if there is still a difference in the number of elements (or only the last element is taken if one of the names only contains one element). Another normalization that has to be done is to switch the order of first and family names. In some languages, it is very common to use the family name followed by a comma and the first
names ("Clinton, William Jefferson" instead of "William Jefferson Clinton"). Here we simply change the order by swapping the strings preceding and following a comma for all names in the database. Note that we swap the strings first and then check for matching number of name elements. As a final preprocessing step, we convert all names to lower case versions to reduce the number of parameters in our models.

Unfortunately, the data set extracted in this way is quite small which is probably due to the requirement of having an info-box (with matching contents). Nevertheless, 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 son of a Jewish family). However, we assume that there are only very few of these exceptions.

From each of the data sets, 50 name pairs were removed to form two test sets for both language pairs. Each of the two test sets is used for evaluating all the models tested. The remaining pairs are used for training and/or tuning model parameters.

5.5.2.2 Evaluation

There are different ways that can be used for evaluating the translations generated. One obvious way is to calculate the accuracy on a test set of unseen names, that is, computing the proportion of correctly translated names in that set. However, accuracy 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. Therefore, other measures for string similarity are considered. We chose to use the Longest Common Subsequence Ratio (LCSR) as our main evaluation measure. It is defined for a pair of strings as the ratio of the length of the longest common subsequence of characters and the length of the longer strings. It produces values between 0 and 1 where 1 is a perfect match between the two strings.

5.5.2.3 Training WFSTs

Parameters for the WFSTs can be trained from data using a forward-backward algorithm. The training set simply contains pairs of correct matching transliterations from the two given
languages and they need not to be aligned. The forward-backward algorithm iteratively maximizes the probability of observing this training data set by adjusting the internal model parameters in a hill-climbing manner. The algorithm converges to a local maximum depending on the initial model chosen.

In our settings we run the training procedure with a uniform initial model and five other randomly chosen initial models. In this way we reduce the likelihood to end up in a suboptimal model at least to some extent. For training, we use Carmel, a free toolkit for manipulating finite state automatons (Graehl, 1997). The training procedure is implemented in the Carmel system and also the procedures for obtaining the most likely strings given some input sequence and model parameters.

As discussed earlier, there are several models that can be applied to our translation task. In particular, the transducer structure in terms of states and their possible emissions can be varied. The first model refers to the edit distance model using separate states for substitutions (and matching); insertion (modeled as inserting target language characters) and deletion (modeled as inserting source language characters). In this model we introduce some kind of model bias by restricting the type of emissions to be of a certain kind at each state. We can also remove this bias by allowing all possible types of emissions (including insertions on the source and target language side) from any state in the WFST. The idea here 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 procedure is capable of learning some underlying structure which is not given to the system when training its parameters. Of course, 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 up 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 five additional randomly chosen initial parameters.

Finally, we can also modify 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. Now we also like to test the technique discussed in section 3 on WFSTs; i.e. splitting words into sequences of vowel/non-vowel N-grams. After splitting our data in this way, we can apply the same training procedures as for the preceding WFSTs.
5.5.2.4 Training PBSMTs

For training and decoding the PBSMT models we used the publicly available toolkit Moses (Hoang et al., 2007) with its connected tools GIZA++ (Och and Ney, 2003) and IRSTLM (Frederico et al., 2008). Adjusting the system to our transliteration task basically refers to manipulating the data sets used for training, tuning and decoding. This means that names have to be split on the character level and spaces have to be converted to some other kind of separation marker (we used '_' for this purpose which otherwise does not appear in our data). Furthermore, we selected monotonic decoding for obvious reasons and left other parameters unchanged. Hence, the model uses standard settings for the alignment with GIZA++ (character alignment in our case), 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) used for tuning. The language model is a 5-gram model estimated from the target language side of our training data using the standard smoothing technique implemented in the IRSTLM toolkit (Witten-Bell smoothing).

There are various fixed parameters that can be tuned in the PBSMT models as described above. 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 experiments we did not tune these training specific parameters. Instead, we concentrated on modifying the 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 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 sections for WFSTs. Here, we do not expect a similar effect on the results as we expect for the WFSTs using this technique. 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 to apply 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 and common character combinations in the target language may not necessarily be as common in transliterated names. 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.

5.5.2.4 Results

Let us first have a look at the baseline for our 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 the unknown words) this is certainly a good strategy if alphabets at least very similar. Hence, the baseline for our task refers to this strategy of copying the strings even for transliterated names. Table 5.4 shows the scores for the baseline in terms of LCSR and accuracy on our test sets of English-Dutch and French-English, each with 50 names. As we can see in Table 5.4, the LCSR scores for both English-Dutch and French-English are quite high already, which means that English and Dutch, or French and English spellings of Russian names are not very far from each other. Even the accuracy is rather high considering the strict nature of this measure. Table 5.5 shows the translation results using our WFST models. As we can see in Table 5.5, the WFST models do not perform very well. None of the Dutch-English WFSTs actually improves the baseline scores, neither in LCSR nor in accuracy. The translation performed seems to actually harm the system especially when looking at accuracy scores. The baseline of leaving names unchanged should be preferred instead. For French-English we can observe a slight improvement of the edit distance WFST. In accuracy, the vowel/non-vowel model also performs better than the baseline. Furthermore, for both language pairs, we can see that the edit distance WFST does not have a clear advantage over a single-state

<table>
<thead>
<tr>
<th></th>
<th>LCSR</th>
<th>acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dutch-English</td>
<td>0.88</td>
<td>0.32</td>
</tr>
<tr>
<td>French-English</td>
<td>0.89</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Table 5.4. Results for baseline approach of copying strings

---

4 We complement LCSR scores with accuracy scores in order to show the magnitude of completely correct transliterations as well
WFST. There is only a slight gain in accuracy for English to Dutch and French to English translation, but otherwise the scores are the same. From our experiments we can also conclude that the training procedure is not capable in learning a hidden underlying structure from the data. However, looking at the size of our training data, this was not to be expected either. Looking at the large differences in scores for various numbers of states it seems that the algorithm easily gets stuck in suboptimal maxima. Finally, the second splitting strategy using vowel and non-vowel sequences does not improve the performance either. On the contrary, it actually hurts the model which is a bit surprising. One reason might be the increased sparseness of our data set including larger sets of input and output symbols, which now contain character N-grams. The only improvements compared to the character-based WFST can be seen in the accuracy for English to Dutch and French to English translations. The score for English-Dutch, however, is still below the baseline.

Let us now look at the results from the PBSMT system in Table 5.6. Here we can see a clear improvement of the translation quality as measured in LCSR scores. Except for the non-tuned models with large language models, all LCSR scores are above the baseline and even accuracy scores exceed the baseline in various cases (but by far not all of them). The importance of training data can be seen in the figures for Dutch and English where the tuning set is included in the otherwise very small training data set. For those experiments, we obtain the highest scores in

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>edit distance</td>
<td>0.88 0.22</td>
<td></td>
<td>0.87 0.20</td>
<td></td>
<td>0.90 0.28</td>
<td></td>
<td>0.89 0.24</td>
<td></td>
</tr>
<tr>
<td>1 state</td>
<td>0.88 0.22</td>
<td></td>
<td>0.87 0.18</td>
<td></td>
<td>0.90 0.26</td>
<td></td>
<td>0.89 0.24</td>
<td></td>
</tr>
<tr>
<td>2 states</td>
<td>0.79 0.00</td>
<td></td>
<td>0.88 0.18</td>
<td></td>
<td>0.79 0.02</td>
<td></td>
<td>0.88 0.14</td>
<td></td>
</tr>
<tr>
<td>3 states</td>
<td>0.81 0.12</td>
<td></td>
<td>0.80 0.04</td>
<td></td>
<td>0.85 0.14</td>
<td></td>
<td>0.81 0.00</td>
<td></td>
</tr>
<tr>
<td>4 states</td>
<td>0.81 0.06</td>
<td></td>
<td>0.85 0.22</td>
<td></td>
<td>0.78 0.02</td>
<td></td>
<td>0.81 0.02</td>
<td></td>
</tr>
<tr>
<td>5 states</td>
<td>0.78 0.02</td>
<td></td>
<td>0.78 0.02</td>
<td></td>
<td>0.79 0.04</td>
<td></td>
<td>0.83 0.04</td>
<td></td>
</tr>
<tr>
<td>vowel/non-vowel</td>
<td>0.83 0.20</td>
<td></td>
<td>0.84 0.28</td>
<td></td>
<td>0.88 0.30</td>
<td></td>
<td>0.87 0.20</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5. WFST Translation results
## Table 5.6 PBSMT Translation results

<table>
<thead>
<tr>
<th>Moses (characters)</th>
<th>Dutch-English</th>
<th>English-Dutch</th>
<th>French-English</th>
<th>English-French</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LCSR</td>
<td>acc.</td>
<td>LCSR</td>
<td>acc.</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>vowel/non-vowel</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>

Both translation directions. For English-French, for which we have a larger training set available, 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 second modification of our training data, the split of characters into vowel/non-vowel sequences, performs quite well, especially for English to Dutch. However, a clear advantage of this technique over the standard pre-processing technique cannot be seen. The final test refers to the inclusion of a larger language model. Here, we included the English, French and Dutch Europarl data [Koehn 2005] for estimating character-based language models. We can clearly see that the additional data sets harm the translation process and only after tuning do the scores 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 data set. This seems to suggest that the overall influence of a language model on translation quality is rather low in our case.

In Table 5.7 some examples of translations from the Dutch/English test set are shown. In these examples we can see typical problems especially of the WFST model. In particular, we can see the problem of consistent character substitutions without considering local context. For example, “i” is consistently translated into “i” in English and “j” into “y” using the WFST. In the
<table>
<thead>
<tr>
<th>Dutch input</th>
<th>Correct English</th>
<th>WFST English</th>
<th>Moses English</th>
</tr>
</thead>
<tbody>
<tr>
<td>andrej tarkovski</td>
<td>andrei tarkovsky</td>
<td>andrey tarkovski</td>
<td>andrey tarkovski</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 solzhenitsyn</td>
<td>alexandr solzhenitsyn</td>
</tr>
<tr>
<td>anton tsjehov</td>
<td>anton chekhov</td>
<td>anton tsyekhov</td>
<td>anton chechov</td>
</tr>
<tr>
<td>Andrej sacharov</td>
<td>Andrei sakharov</td>
<td>Andrej sakharov</td>
<td>Andrei sakharov</td>
</tr>
<tr>
<td>Dmitri sjostakovitsj</td>
<td>Dmitri shostakovich</td>
<td>Dmitri syostakovitsy</td>
<td>Dmitri sjostakovitsj</td>
</tr>
<tr>
<td>Leonid brezjnev</td>
<td>Leonid brezhnev</td>
<td>Leonid brezynev</td>
<td>Leonid bruzhnev</td>
</tr>
</tbody>
</table>

Table 5.7. Examples from the test set Dutch-English showing some typical problems of translating transliterations with the models

PBSMT model, contextual dependencies are better covered due to character n-grams in the translation table. However, there are still ambiguities causing problems in examples like “tsjehov”→“chechov” (instead of “chekhov”).

5.5.2.4 Conclusions

Spellings of names originally coming from a different writing system vary substantially even for related languages. Two types of models have been investigated for translating such names between Dutch-English and English-French. Various types of WFSTs and PBSMT models were applied to this task. The models were trained on name pairs of Russian origin extracted from Wikipedia. Results show the PBSMT system performing best, consistently beating the baseline. These results are encouraging especially when considering the tiny training set that was available to us. The results also show that specialized models like the ones that have been tested for generating transliterations may help to handle unknown words and hence improve performance in NLP applications such as MT, CLIE, and CLIR. As future work it should be interesting to extend this work from pair HMM to more general DBN models in which issues such as context are automatically modeled.
Bibliography


Appendices

Appendix A: Publications, Conference Presentations, and Research Talks

Publications


Submitted


Conference Presentations

1. Peter Nabende and Jörg Tiedemann. An Evaluation of Phrase-based SMT and Finite State Transducer models for Translating Transliterations, Accepted for Presentation at the 30th TABU Dag, University of Groningen, Netherlands, June 2009


Poster Presentations


Research Talks


5. Also made a presentation in a seminar organized by Dr. John Quinn at FCIT, Makerere University concerning the application of HMMs for Machine Transliteration with respect to my research in Computational Linguistics Group, Alfa Informatica.
**Appendix B: PhD Research schedule**

**Registration period:** 06/10/2007 – 06/10/2011 (4 Years)

<table>
<thead>
<tr>
<th>Start date – End date</th>
<th>Duration (months)</th>
<th>Location</th>
<th>Specific Task</th>
</tr>
</thead>
</table>
| 3/03/2008 – 13/12/2008| 9.25              | RuG, Netherlands  | - Literature review on machine transliteration models [Reading group presentation – 14/04/2008]  
|                       |                   |                   | - Pilot experiments on using pair-HMM for estimating the similarity between entity names for languages of the same alphabet (English – French) [TABUDAG meeting presentation – 6/06/2008]  
|                       |                   |                   | - Review on EM algorithms and estimating the parameters of a pair-HMM [July-August]  
|                       |                   |                   | - Modification of pair-HMM software to use two alphabets for English-Russian datasets [09-10/2008]  
|                       |                   |                   | - Literature review on translating transliterations [ICCIR 2009 Paper – 10/10/2008]  
|                       |                   |                   | - Dutch language course and C++ programming language course [08-12/2008]  
|                       |                   |                   | - PairHMM experiments for English-Russian dataset with distinct alphabets [CISSE conference – 10/12/2008]  
|                       |                   |                   | - Data extraction and pair-HMM experiments for extracting English – Russian named entities from Wikipedia  |
| 14/12/2008 – 4/01/2009| 0.75              | Mbale, Uganda     | - Work on pair-HMM experiments for extracting bilingual entity names from Wikipedia. Training data and evaluation data.        |
|                       |                   |                   | - Attended MaxEnt model tutorial [12-16/01/2009]  
|                       |                   |                   | - Attended LOT Winter School [19 – 30/01/2009]  
|                       |                   |                   | - Participated in 19th CLIN meeting [22/01/2009]  
|                       |                   |                   | - Attended TLT workshop. [24/01/2009]  
|                       |                   |                   | - Made poster presentation during LOT Winter school poster session [26/01/2009]  |
|                       |                   |                   | 02/2009                                                                      |
|                       |                   |                   | - Review Dynamic Bayesian Networks (DBNs) (for Alfa Informatica reading group presentation)  
|                       |                   |                   | - Review similarity estimation techniques [for evaluation against pair-HMM estimation when applied on identification tasks for different writing systems]  |
- Attended statistics seminar for Wednesdays (16 – 18 hrs) and C++ programming language course for Thursdays (9 – 11 hrs)

**03-04/2009**

- Participated in ACL-IJCNLP Named Entity Workshop shared task on transliteration:
  - Applied parameter estimates from pair-HMM training to generating transliterations between English and Russian.
  - Made two reading group presentations concerning Dynamic Bayesian Networks.
  - Made a presentation in a statistics seminar associated with using Cross Entropy for evaluating transliteration quality from different models

**05-07/2009**

- Participated in Belgian – Dutch Conference on Machine Learning (Benelearn 09)
- Final Write up of First year Report
- To participate in 30th TABU Dag in June.
- Carry out Experiments associated with applying Dynamic Bayesian Network models to estimating similarity between candidate transliterations
- Extend work on recognizing bilingual entity names from Wikipedia
- Develop a transliteration generation component for use with DBNs.
- Evaluate more techniques against the pair HMM for estimating similarity between transliterations

<table>
<thead>
<tr>
<th>1/08/2009 – 1/01/2010</th>
<th>5</th>
<th>Makerere, Uganda</th>
</tr>
</thead>
</table>

**08-12/2009**

- Plan to participate in ICCIR’09 conference [2-5/08/2009]
- Plan to participate in ACL-IJCNLP NEWS 2009 Workshop in Singapore [6-7/08/2009]
- Plan to teach a course in semester 1 [08-12/2009] at Makerere University
- Do more experimental work associated with bilingual named entity recognition and transliteration generation.
- Thesis write-up (introductory chapters including experiments)
Continue with Literature review on:
  - Bilingual named entity recognition and machine transliteration techniques
  - Similarity estimation techniques

<table>
<thead>
<tr>
<th>2/01/2010 – 30/06/2010</th>
<th>6</th>
<th>RuG, Netherlands</th>
</tr>
</thead>
</table>

- Additional experiments on tasks that need recognition and transliteration of entity names.

65
<table>
<thead>
<tr>
<th>Date Range</th>
<th>Duration</th>
<th>Location</th>
<th>Activity Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/07/2010 – 30/10/2010</td>
<td>4</td>
<td>Makerere, Uganda</td>
<td>- Continue with Literature Review, experiments, and Thesis Write up</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Draft Thesis</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>48</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>