Comparison of applying Pair HMMs and DBN models in Transliteration Identification

Peter Nabende

Alfa-Informatica, University of Groningen
p.nabende@rug.nl

Abstract

Transliteration is aimed at dealing with unknown words in Cross Language Information Retrieval (CLIR) and Machine Translation (MT). Most of the transliteration tasks depend on a similarity estimation stage where a model is utilized with the aim of identifying a transliteration match for a given source word. In this paper, we evaluate the application of two related frameworks to transliteration identification. Both frameworks model string similarity as the cost incurred through a series of edit operations. One framework implements Pair Hidden Markov Models (Pair HMMs) (Mackay and Kondrak 2005) while the other implements classes of Dynamic Bayesian Network (DBN) models (Filali and Bilmes 2005). For each Pair HMM, we adapt different algorithms for computing transliteration similarity estimates. For the DBN framework, we modify the DBN classes in (Filali and Bilmes 2005) and specify models from the classes to represent factorizations that we hypothesize could affect the value of a transliteration similarity estimate. Separate tests applying models from the two frameworks result in high transliteration identification accuracy on an experimental setup of Russian-English transliteration. A check on the output from models associated with the two frameworks suggests that there can be improved transliteration identification accuracy through a combination of models.

1 Introduction

Transliteration tasks require analyzing strings where each of the languages uses a different writing system, for example determining the level of similarity between a string written in English "Czeladź" and its Russian representation "Челядзь". The main aim of a transliteration analysis process is to determine a correct representation of a string in a different writing system that is expected to bear a pronunciation similar to that of the original source word. It is no surprise that most of the earliest attempts at automated transliteration proposed phoneme-based approaches (Knight and Graehl 1998, Jung et al. 2000). Recently, approaches that consider only orthographic representations have resulted in comparable if not better machine transliteration performance than phoneme-based approaches (Li et al. 2004). Currently, various approaches are being sought to develop automated transliteration systems; the shared tasks on machine transliteration and transliteration mining (Li et al. 2009, Kumaran et al. 2010) represent such ongoing attempts at evaluating state of the art machine transliteration systems. In this paper, we investigate the use of two methods in automated transliteration identification: Pair HMMs and DBNs.

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

The two methods model transliteration as a cost incurred in transforming a source word to a target word through a series of edit operations. The three edit operations are: substitution, insertion, and deletion. Each of the methods utilizes different algorithms for estimating the edit cost as a similarity score from the edit based transformation of a source string to a target string. The edit cost is in turn used to find the most likely transliteration out of a set of candidate transliterations. The Pair HMM method specifies the edit operations as states in which we estimate parameters associated with source-target character relationsips including those with empty symbols. The DBN framework specifies source-target string transformation through factorizations on edit operations. The two methods have been proposed in previous related work (Filali and Bilmes 2005, Mackay and Kondrak 2005), and although there are comparisons when applied in the task of cognate identification (Kondrak and Sherif 2006), the proposal to apply the methods to transliteration tasks, also necessitates comparison. In this paper, we specify and test additional variants of models from both frameworks and later compare their performance. The models from both frameworks are also evaluated against those of a baseline method of using paired trigram statistics in the given source and target language corpora. We also check the transliterations identified from each model with the aim of determining whether there could be benefit in combining two or more models from each framework or from both frameworks for estimating transliteration similarity. The paper is organized as follows: section 2 describes the Pair HMM and DBN frameworks, section 3 suggests some of the challenges associated with using the two frameworks on data associated with different writing systems, section 4 describes the transliteration experimental setup and discusses results from testing different pair HMM and DBN models. Section 5 concludes the paper with pointers to future work.

2 Models for Transliteration similarity estimation

Transliteration similarity estimation can be looked at as determining the level of relationship between two strings in different writing systems. Different methods can be used in the process of measuring string similarity. A common approach that is followed in this paper uses the notion of edit distance where string similarity is associated with the cost of 'edit operations' required to transform one string to another string. Figure 1 illustrates the sequence of edit operations required to transform the Russian string "пётр" to the English string "Peter" (a) and the Dutch string "Pieter" (b). In Figure 1, the edit operations are denoted by M(Substitution), and I(Insertion). For the case where we would have aligned a character in the Russian string to a gap in the English or Dutch string, the edit operation is denoted as a D(Deletion). The Pair HMM and DBN methods represent this process differently, and we distinguish between them in the following subsections.

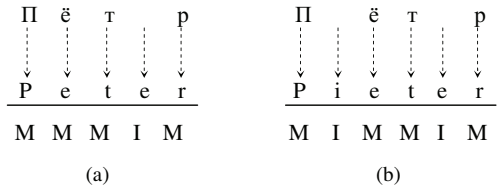

Figure 1: Illustration of alignment through edit operations required to transform a name written in Russian "пётр" using the Cyrillic alphabet with corresponding English (a) and Dutch (b) representations written using the Latin alphabet.

2.1 Edit distance based Pair HMM method

In a Pair HMM, edit operations such as those illustrated in Figure 1 are defined as "emission" states that are used to model the relationship between source and target language characters. The probabilistic relationship for each source language character and target language character is modeled in the substitution state (M); that for each source language character and an empty target language symbol is modeled in the deletion state (D); and that for each each target language character and source language empty symbol is modeled in the insertion state (I). We also have the option to represent transitions between a Pair HMM's states in various ways. A transition parameter represents the probabilistic value associated with moving from one state to another or the same state for a given Pair HMM. While using Pair HMMs, the main focus is usually on comparing the effectiveness of different Pair HMM algorithms and determining the optimal structure of the underlying model. Generally, to determine the optimal Pair HMM structure, we can examine the relative contribution of three sets of parameters (Mackay and Kondrak 2005): substitution parameters, gap parameters (insertion and deletion), and transition parameters. Because substitution parameters constitute the core of a Pair HMM, focus is usually put on the gap and transition parameters. In this paper, we determine the effect of Pair HMM transition parameters on the task of transliteration similarity estimation for a given language pair dataset. We have therefore specified four Pair HMM variants where we vary the size and properties of transition parameters.

The first Pair HMM variant does not use transition parameters between each of the edit states; it only uses transition parameters from a start state to one of the edit states, and from one of the edit states to the End state. The second Pair HMM variant (Figure 2(a)) uses three transition parameters (α, β, and δ), where each transition parameter is associated with leaving one of the edit states and the starting parameters are associated with the transition parameters from the substitution state to one of the edit states. The third Pair HMM variant (Figure 2(b)) is adapted from previous work (Mackay and Kondrak 2005, Wieling et al. 2007) where the starting parameters are also associated with the transition parameters from the substitution state. The last Pair HMM variant (Figure 3) uses nine distinct transition

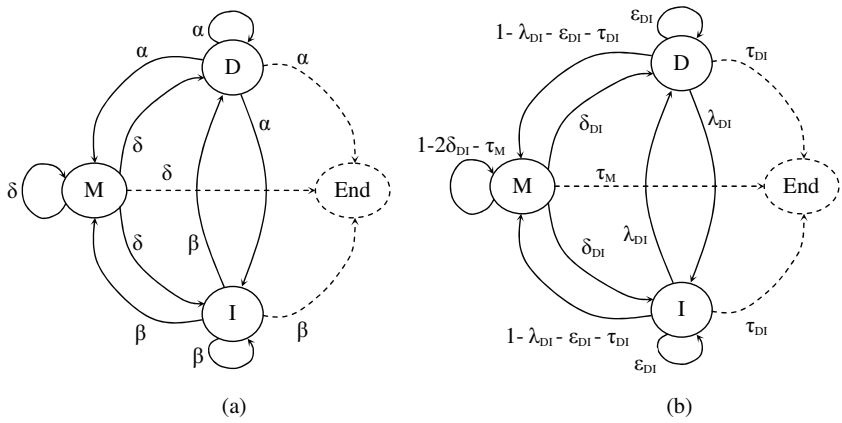

Figure 2: (a) Pair HMM with three transition parameters (b) Pair HMM with five transition parameters (Mackay and Kondrak 2005). Solid edges are associated with transitions to emitting states and solid nodes are emitting states. dotted edges transit to the non-emitting End state.

parameters between each of the Pair HMM states and the starting parameters have similar properties as those of the two previous Pair HMM variants. Each variant is designed to model the parameters in the insertion and deletion states distinctly.

For each Pair HMM variant, an implementation of the Pair HMM's Baum-Welch algorithm is used to estimate all its transition and emission parameters given source target language training data. Four different algorithms are defined for each Pair HMM variant and are used to compute similarity scores for source and target language strings given the corresponding Pair HMM. The scoring algorithms are (Mackay and Kondrak 2005): forward, Viterbi, Viterbi log odds, and forward log odds. For a brief description of these algorithms: the forward algorithm considers all possible alignments when determining a string similarity estimate; the Viterbi algorithm considers the best alignment(s); and the log odds versions normalize the base algorithm (forward or Viterbi) scores using a random model score. The random model assumes no relationship between sequences and captures the probability of a pair of symbols co-occuring by chance.

2.2 DBN-based edit distance method

The DBN-based edit distance method uses the graphical models approach of Dynamic Bayesian Networks where random variables are used to represent hidden edit operation states that are used in estimating string similarity. Generally, DBNs are used to represent both time-series data that is generated by some causal process and sequence (e.g. Natural Language Processing or Biological) data where

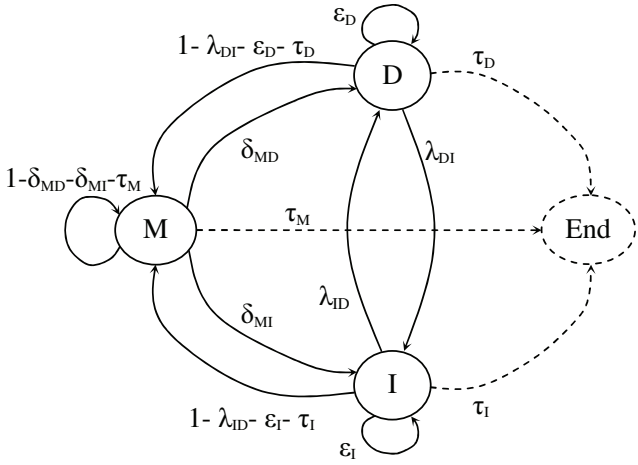

Figure 3: Pair HMM with nine transition parameters.

we are doubtful about the generating mechanism (Murphy 2002). DBNs generalize HMMs by representing the hidden state as an arbitrary set of random variables with arbitrary conditional independence assumptions (Zweig and Russell 1998).

The DBN method is implemented using the Graphical Modelling ToolKit (GMTK) (Bilmes and Zweig 2002) that simplifies the representation and modification of various types of models including HMMs. We have various options to vary the type of dependencies on the hidden edit operation states representing various factorizations that we postulate could affect the probabilistic similarity value associated with a pair of strings. Most of the models that were initially specified in (Filali and Bilmes 2005) are adapted in this paper. We start by adapting the base DBN model that is also referred to as the Memoryless and Context Independent (MCI) model. Figure 4 is shows the graphical templates for the start Bayesian Network (BN), *inter-slice* BN, chunk BN, and end BN for the MCI DBN model.

To help understand the graphical representations of the edit distance based DBN models, let us first review the stochastic extension of string edit distance in (Filali and Bilmes 2005) upon which the edit distance based DBN framework was developed. Given a source string $s_1^m = s_1 s_2 ... s_m$ of length *m* over an alphabet A_s, and a target string t_1^n of length *n* over an alphabet A_t; we can model edit operations using a hidden random variable Z, that takes values in $(A_s \cup \epsilon \times A_t \cup \epsilon) \setminus (\epsilon, \epsilon)$ where ϵ represents an empty string, and *Z* is perceived as a random vector with two components ($Z^{(s)}$ and $Z^{(t)}$). To estimate string similarity, we follow a similar approach, where we determine the joint probability $P(s_1^m, t_1^n|\theta)$ of observing the source/target string pair (s_1^m, t_1^n) given model parameters θ. In (Filali and Bilmes 2005), the probability of a particular pair of strings is expressed as the sum of the probabilities of all possible ways of generating the pair:

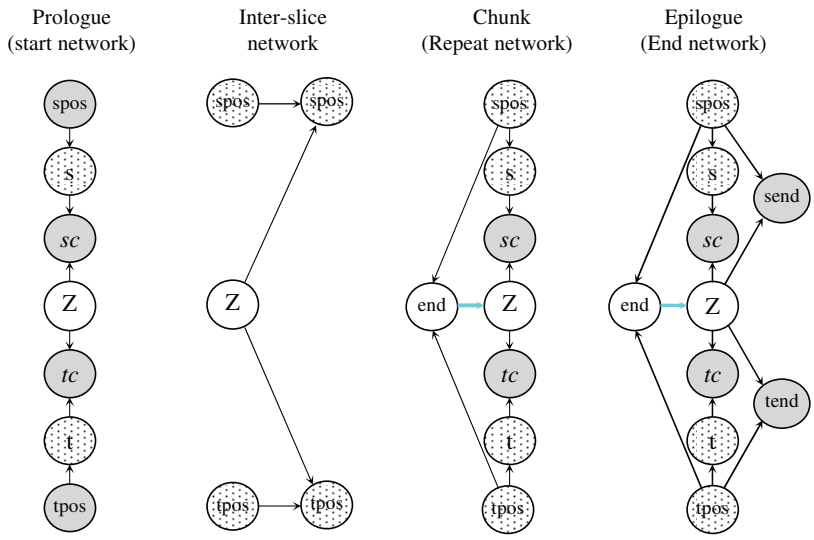

Figure 4: Graphical template for the MCI DBN model. Following the common convention for representing graphical models, shaded nodes represent observed variables, unshaded nodes represent hidden nodes, and nodes with dots represent deterministic hidden variables. Adapted from (Filali and Bilmes 2005).

$$P(s_1^m, t_1^n|\theta) = \sum_{z_1^l : v(z_1^l)=\langle s_1^m, t_1^n\rangle, max(m,n)\leq l\leq m+n} \sum P(z_1^l, s_1^m, t_1^n|\theta)$$

where $v(z_1^l)$ represents the string pair generated from the sequence z_1^l.

For the MCI DBN model, because there is no dependence between edit operations, $P(z_1^l, s_1^m, t_1^n|\theta)$ can be factored as $\prod_i P(z_i, s_1^m, t_1^n)$ where $1 < i < l$ and $z_i = \langle z_i^{(s)}, z_i^{(t)}\rangle$. It is also noted that although the term *context independent* is used for the MCI DBN model, there is a global dependence between z_i and source target string symbols that forces z_1^l to generate (s_1^m, t_1^n).

In Figure 4, Z represents the current edit operation variable. *spos* and *tpos* are variables representing the current position in the source and target strings respectively. *s* and *t* represent the current character in the source and target strings respectively. The *sc* and *tc* nodes are source and target consistency[1] nodes. The end node is a switching parent of Z and represents the variable that indicates when

[1]The *sc* and *tc* nodes have a fixed observed value 1 and the only configuration of their parents are such that the source component of the edit operation variable Z is *s* or an empty symbol for *sc* and t or an empty symbol for *tc* and that Z does not generate empty source and target symbols at the same time (Filali and Bilmes 2005)

we are past the end of both the source and target strings, i.e. when $spos > m$ and $tpos > n$. The *send* and *tend* nodes represent variables that ensure that we are past the end of the source and target strings respectively.

Based on the MCI model, we have currently adapted three other DBN model classes that were initially introduced in (Filali and Bilmes 2005). The DBN model classes represent different dependencies on the edit operation random variables and include: edit operation memory dependencies; source and / or target character context dependencies; and edit operation length dependencies. We briefly point out the main properties of these other DBN models.

Figure 5 represents the interslice network for a context independent Memory (MEM) DBN model. The starting, chunk, and end networks for the MEM DBN model are not included in Figure 5 since they are similar to those of the MCI DBN model (Figure 4). In Figure 5, a variable H that can be used to implement various dependencies with Z_i is introduced. Generally, H can be stochastic or deterministic, and its cardinality determines the amount of information that can be "summarized" from one slice to another (Filali and Bilmes 2005). In this paper,

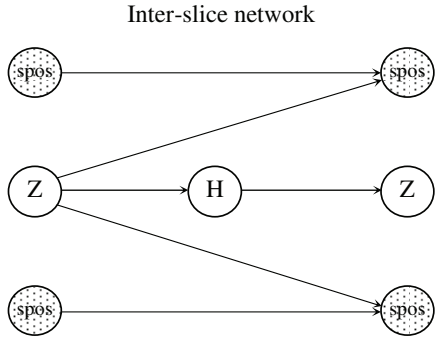

Figure 5: Inter-slice graphical template for the context independent Memory model. Adapted from (Filali and Bilmes 2005).

we only test the deterministic implementation of H, where H is a simple copy of Z, which is as well represented by the specification $P(Z_i|Z_{i-1})$.

Figure 6 represents the starting, inter-slice, chunk, and end networks for a Context (CON) dependent DBN model, where we add edges from s_i, $sprev_i$ to Z_i. In the other CON DBN model that is tested, we only add an edge from s_i to Z_i.

Figure 7 shows the graphical templates for the inter-slice and intra-slice networks for the MCI length DBN model. Additional variables are used to specify the logic needed to factor in the length of the edit sequence in the final transliteration similarity estimate. In Figure 7, *incl*, represents a random variable that determines the number of allowed edit operations. The variable *cnt* is used to determine the slice number and is used to trigger the random variable *reql* when the required frame number is reached.

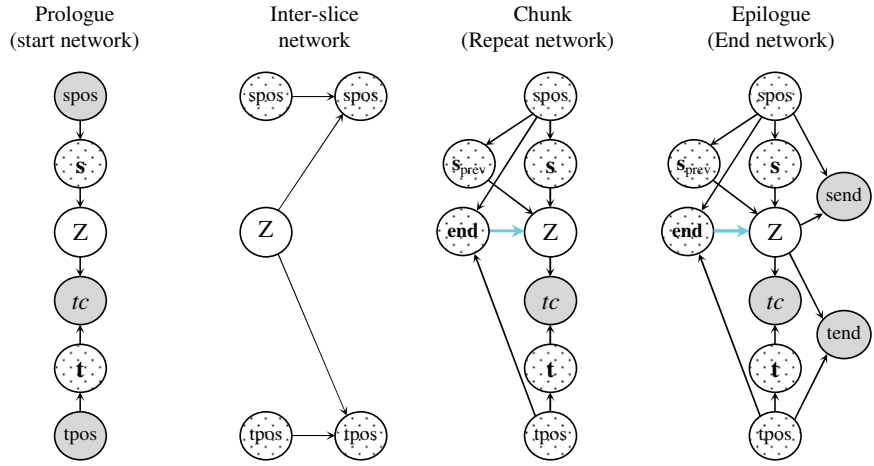

Figure 6: Graphical template for a context dependent DBN model.

We mainly modify the DBN models considering the type of data on which they are to be applied. For example in previous work, the DBN models were applied on data where the source and target language used the same writing system and hence the size and alphabets that were specified for both languages was the same. For the transliteration task, the source and target languages use many different characters and the total number of characters used for each writing system is not the same; we therefore modify the DBN models to represent the different characteristics inherent in the source-target language transliteration data.

For each DBN model, the training procedure requires and leads to the specification of various types of parameters including: a *triangulated* structure for the DBN model, conditional probability tables representing the dependencies between variables for both *inter-slice* and *intra-slice* networks for a given DBN; decision trees; deterministic relationships; etc. A generalized Expectation Maximization (EM) algorithm is used for each of the DBN models that are tested in this paper. Similar to the cases in (Filali and Bilmes 2005, Kondrak and Sherif 2006), we use only three iterations for the EM procedure to avoid over-fitting the models. We have observed this number of EM iterations to be optimal for transliteration data as well. To obtain similarity estimates for source target language strings, we use GMTK's junction tree algorithm which uses learned parameters from the EM step and also uses transliteration data dependent parameters.

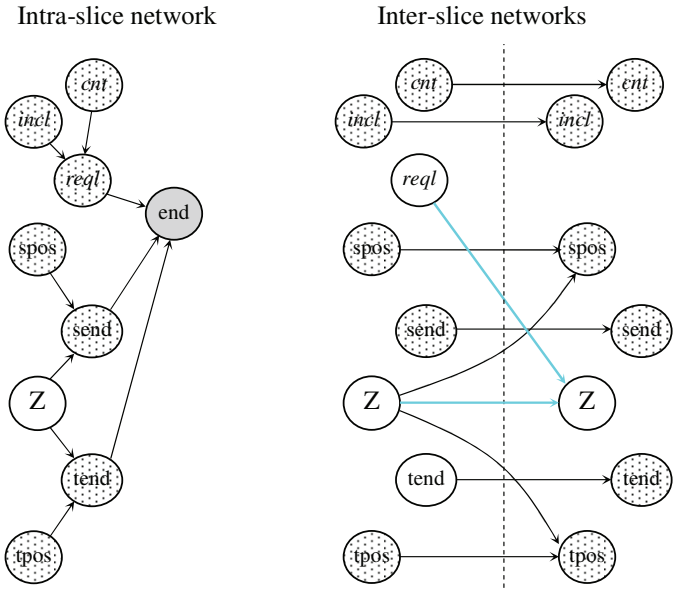

Figure 7: Graphical template for the MCI length DBN model. Some nodes and edges for example those associated with source and target symbols are omitted in this figure to simplify the description of additional variables for factoring in the edit sequence length. Adapted from (Filali and Bilmes 2005).

3 Challenges of applying edit distance based methods to transliteration

For the edit distance based methods, we specify tokens on single character basis. To improve processing, each token takes the form of a unique whole number. For example, if English is one of the languages and 26 characters are used in the data, each of the characters is converted to a whole number wn_i, where $0 \leq wn_i \leq 25$. This kind of approach to character representation is expected not to be suitable for all writing systems. Previous work (Filali and Bilmes 2005, Mackay and Kondrak 2005, Kondrak and Sherif 2006, Wieling et al. 2007, Nabende et al. 2010) suggests that tasks where languages use phonemic alphabets (for example the Latin alphabet) usually benefit from this approach. As a preliminary test, we applied the Pair HMM variant illustrated in Figure 3 on UTF-8 encoded Chinese-English transliteration data from the NEWS 2009 shared task on machine transliteration, and where Chinese is analyzed as the source language while English is the target language. In this case the writing system used for Chinese is mostly different from that of a phonemic alphabet and is mainly considered to be syllabic or logographic. For the Chinese-English transliteration data, 26 characters are used for the English part of the dataset and 370 simplified Chinese characters are used for the Chinese part of the dataset. We mapped each of the characters to unique whole numbers

in each of the respective languages and carried out an experimental transliteration identification run. We trained the Pair HMM variant on 31961 matching Chinese-English name pairs. We then tested the model on 2896 Chinese names in a held out test set. We obtained an accuracy of 0.213 and a Mean Reciprocal Rank (MRR) of 0.327 for the forward log odds algorithm which resulted in the best performance in this case. These preliminary results on the Chinese-English transliteration data suggest that the edit distance based methods that are tested in this paper are not valuable for all writing systems unless when a suitable representational approach is used. We intend to revisit this issue as future work where we propose to apply the Pair HMMs on a Romanized representation of the Chinese transliteration data.

4 Experiments

Two sets of experiments have been used: one set for Pair HMMs and the other for DBN models. The main difference in the two sets of experiments is the approach that is used to evaluate the models. A *stratified 10-fold cross validation* approach is used for the Pair HMMs while a *hold-out* method is used for the DBN models. The difference in testing approaches is mainly attributed to the amount of time it takes for some of the DBN models to execute to completion on the transliteration data-sets. The Pair HMMs are relatively faster on the same data-sets making it easy to follow the cross-validation approach. For both sets of experiments, UTF-8 encoded Russian-English datasets associated with the 2009 shared task on Machine Transliteration are used. When comparing models from each of the edit distance based framework, we use results from the respective sets of experiments. However, when comparing models across the two frameworks, only the *hold-out* method is used to enable evaluating more plausible DBN models.

4.1 Pair HMM Experiments

For the Pair HMMs, we split the English-Russian datasets into ten subsets that are mutually exclusive on the test set and where the set of characters used is the same as in the original dataset. Therefore, from a total of 7840 Russian-English transliteration pairs, we have 784 names in each subset as test data and the remaining data as training data. On the Russian side of the dataset, 34 characters constitute the alphabet while for English, 86 characters consitute the alphabet used for English. The large size of characters for English are mainly due to the large number of diacritics that are used for English in this particular dataset. We leave any diacritics in both datasets so as to maintain any associations that may exist regarding their rendering the other language.

The Cross Validation Accuracy (CVA) of any Pair HMM algorithm that is tested is simply calculated by averaging 10 individual accuracy measures:
$CVA = \frac{1}{10}\sum_{n=1}^{10} A_n$, where A_n is the accuracy of a model on the n^{th} test dataset,

$A_n = \frac{1}{N}\sum_{i=1}^{N} 1$ if $\exists r_{i,j} = c_{i,1}$; 0 otherwise, where $r_{i,j}$ is the j^{th} reference transliteration for the i^{th} name in the test set and $c_{i,1}$ is the first candidate transliteration that is identified based on a model's similarity estimate. N is the total number of names in each test set.

In a manner similar to that for CVA, the Cross Validation Mean Reciprocal Rank (CVMRR) is calculated by averaging the 10 individual MRR measures: $CVMRR = \frac{1}{10}\sum_{n=1}^{10} MRR_n$, where MRR_n is the MRR of a model on the n^{th} test set, $MRR_n = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{R(i)}$, where $R(i)$, is the rank of the correct matching transliteration candidate in a set of identified candidate transliterations associated with the i^{th} source language name in the test dataset. The values for CVA and CVMRR range from 0 to 1 and the closer the CVA or CVMRR is to 1, the better is a model's transliteration identification quality.

The CVA and CVMRR results for the different Pair HMM variant algorithms are shown in Table 7.1. The results suggest that it is important to model for Pair HMM transition parameters. The results show that transliteration identification accuracy improves when we increase the number of plausible transition parameters. However, there is only a slight improvement in accuracy for three algorithms of the Pair HMM variant with nine transition parameters (phmm_9_trans) compared to that with five transition parameters (phmm_5_trans). The results also suggest that the forward algorithm for each Pair HMM variant perform much better than the other algorithms. These transliteration identification results are different from those in (Mackay and Kondrak 2005) where the log odds algorithms performed best from the cognate identification task. As we reported from our preliminary test run on Russian-Chinese transliteration data, the results suggest that specific language pairs influence the relative performance of the Pair HMM algorithms.

4.2 DBN Experiments

For the DBN experiments, we use the first subset of data from the Russian-English transliteration data described in section 4.1. Excluding the division of the data into 10 subsets, the dataset is mantained as described in section 4.1. Since the *hold-out* method is used, the DBN models are evaluated using Accuracy and MRR for this particular dataset. In the next section, the results for the DBNs are presented against those for the Pair HMMs on the same dataset.

4.3 Results from applying DBN models and best Pair HMM algorithms

The evaluation results for the DBNs against the Pair HMMs are shown in Table 7.2. Table 7.2 also shows results from a *pair n-gram* (in this case *tri-gram*) that is used as a baseline. The *pair tri-gram* model uses counts based on the occurrence of source-target trigrams. The method works in such a way that the source and target

Pair HMM Variant	Scoring Algorithm	CVA	CVMRR
phmm_0_trans	Forward	0.9174	0.9473
	Viterbi	0.7958	0.8228
	Forward log odds	0.4189	0.5064
	Viterbi log odds	0.5296	0.6358
phmm_3_trans	Forward	0.9272	0.9522
	Viterbi	0.9222	0.9397
	Forward log odds	0.8906	0.9213
	Viterbi log odds	0.8003	0.8464
phmm_5_trans	Forward	**0.9806**	**0.9864**
	Viterbi	0.9302	0.9443
	Forward log odds	0.9159	0.9468
	Viterbi log odds	0.8343	0.8764
phmm_9_trans	Forward	**0.9798**	**0.9858**
	Viterbi	0.9342	0.9477
	Forward log odds	0.9208	0.9506
	Viterbi log odds	0.8451	0.8853

Table 7.1: CVA and CVMRR results for the Pair HMM variant algorithms. phmm_0_trans denotes the Pair HMM that uses no transition parameters; phmm_3_trans denotes the Pair HMM with three transition parameters (Figure 2(a)), phmm_5_trans and phmm_9_trans denote Pair HMMs with five transition parameters (Figure 2(b)) and nine transition parameters (Figure 3) respectively. The highest CVA and CVMRR values are in bold.

language strings are divided in overlapping *tri-grams* and the *tri-grams* are linked based on their position in the string. Related *tri-grams* are counted and the counts are used in estimating the similarity between two strings. In Table 7.2, we show results for the best performing Pair HMM algorithms on this dataset for each Pair HMM variant. phmm_0_trans, phmm_3_trans, phmm_5_trans, phmm_9_trans are as defined in Table 7.1 above. dbn_mci represents the base MCI DBN model (Figure 4); dbn_mci_len represents the MCI length DBN model (Figure 7); dbn_mem represents the MEM DBN model (Figure 5); dbn_con_s_i_s_{i-1} represents a CON DBN model where the edit operation random variable Z depends on the current (s_i) and previous (s_{i-1}) characters in the source string (Figure 6); dbn_con_s_i represents a CON DBN model where Z has a contextual dependency on the current character (s_i) in the source string; and dbn_con_s_i_len represents a CON length DBN model that factors in the dependency of Z on the current character (s_i) in the source string and the length of edit operations for computing the string similarity estimate.

Training times for the respective models are also included in the second column in Table 7.2. As shown in Table 7.2, the baseline model and Pair HMMs take shorter times for training, and the same is true for computing transliteration pair similarity values. Moreover, the training times for the Pair HMMs represent

Model	**Training Time (mins)**	**Accuracy**	**MRR**
baseline(pair trigram)	2.5	*0.9270*	*0.9450*
phmm_0_trans	< 3	0.9209	0.9501
phmm_3_trans	< 3	0.9401	0.9577
phmm_5_trans	< 3	0.9834	0.9898
phmm_9_trans	< 3	0.9834	0.9897
dbn_mci	146	0.9783	0.9853
dbn_mci_len	140	0.9031	0.9408
dbn_mem	34	0.8876	0.9210
dbn_con_s_i_s_{i-1}	49	0.9796	0.9834
dbn_con_s_i	22	**0.9872**	**0.9906**
dbn_con_s_i_len	21	0.9834	0.9887

Table 7.2: Accuracy and MRR Results from applying best Pair HMM variant algorithms and DBN models on 784 Russian-English test name pairs.

the total time over hundreds of iterations using the Buam-Welch algorithm on the Russian-English transliteration datasets. The DBNs on the other hand take relatively longer to train for the three specified EM iterations on the same transliteration dataset. It also takes longer while using DBN models in computing transliteration similarity estimates over the test dataset. However, the generic nature of the DBN framework guarantees a bigger model space for exploration, and there are already successful attempts at improving computational efficiency using DBNs.

For the qualitative measures, most of the edit distance based models perform better than the baseline method of pair trigrams apart from the dbn_mem and phmm_0_trans models. The results associated with the DBN models in Table 7.2 to a large extent are similar to those that were obtained from the pronunciation classification task in (Filali and Bilmes 2005). The results also show that the DBN context dependent models perform better than other DBN models. We also see that the context dependent DBN model (dbn_con_s_i) performs best overall although only slightly better than some of the best performing Pair HMMs and DBN models. The DBN MCI model, surprisingly performs better than some complex DBN models which is unlike the case in (Filali and Bilmes 2005). But this could be becuase of the nature of the English-Russian datasets in which both languages use a phonemic alphabet and a one to one character mapping may be already sufficient to compare a pair of strings. The results in Table 7.2 also suggest that additional information in some cases slightly lowers the transliteration identification accuracy of the DBN models on this particular dataset: for example attempting to factor information about the size of edit operations in a similarity estimate slightly lowers the model's transliteration identification accuracy although to an insignificant level. But this is also the case with the Pair HMMs where we have significantly better performance with the Pair HMM base Forward and Viterbi

algorithms as compared to the log odds algorithms that involve using additional information from a random model. The context dependent DBN models generally perform better, underlining the need for contextual representation in transliteration similarity estimation.

We have also looked through the transliteration identification output from some of the models. Table 7.3 shows the case where two models: a context dependent length DBN model and a Pair HMM forward algorithm did not provide the correct transliteration match at the first rank. A check on the output of the Pair HMM algorithms showed that they were failing on the same source name(s) while that for DBNs showed that they were mostly failing on different source names. Comparing the output of some of the Pair HMMs and the DBNs showed that a Pair HMM failed on many if not completely different source names when compared with a DBN model as is the case in Table 7.3. If the two models being compared were combined, the results show that the combination would have resulted in 100% transliteration identification accuracy. This seems to suggest that combining models from the two edit distance based methods could significantly improve transliteration identification accuracy. However, the challenge remains in deciding on what similarity estimate to use knowing that it is not yet possible to evaluate the two methods at the stage when they only provide a similarity estimate.

Context dependent DBN model			**Pair HMM(Forward algorithm)**		
Russian	**English**	**Rank**	**Russian**	**English**	**Rank**
Торунь	Toruń	34	Бештау	Beshtau	770
Спика	Spica	2	Ицпапалотль	Itzpapalotl	2
Бюзум	Büsum	2	Вайльтинген	Weiltingen	4
Фанагория	Phanagoria	5	Кураш	Kurash	2
Млынары	Młynary	2	Бернтайленд	Burntisland	2
Биттерфельд	Bitterfeld	12	Иккермюнде	Ueckermünde	2
Раштатт	Rastatt	2	Пролетариат	Proletariat	2
Эльстерауэ	Elsteraue	3	Файтсбронн	Veitsbronn	2
Трапштадт	Trappstadt	2	Валлетта	Valletta	2
Давидсон	Davidson	2	Плеоназм	Pleonasm	7
Куйбышев	Kuybyshev	596	Секстант	Sextant	2
Эгвекинот	Egvekinot	294	Радельфинген	Radelfingen	4
Тепейоллотль	Tepeyollotl	2	Арзамас	Arzamas	2

Table 7.3: Examples of where two models associated with the two edit distance based methods lead to identification of correct transliterations at ranks other than first rank. For these two models the table shows that they fail on completely different observation sequences and if combined could significantly complement each other.

In Table 7.3, we also observe some level of consistency in failing to identify strings at the first rank. For example we see that in most cases, where the Russian soft sign "ь" appears after one of the Russian characters (for example л 'l'), the

correct match is in most cases returned beyond the 2^{nd} rank.

5 Conclusion and Future Work

We have evaluated Pair HMMs and DBN models on an experimental setup of a transliteration identification task. We have shown that in their status as reported in this paper, models from the two frameworks lead to high transliteration identification accuracy. But also by looking at the identification results for each of the models, we propose that designing language dependent post-processing procedures after applying the edit distance based models could improve transliteration identification accuracy. Results from applying models in both frameworks also suggest that a combination of models would also lead to improved transliteration identification accuracy.

As future work, based on the results, we propose an investigation into ways of combining models from the two edit distance based methods methods for transliteration mining and transliteration generation. We also plan to investigate how models from the two edit distance based methods could be applied to other writing systems (for example Chinese) so as to result in improved transliteration identification.

Acknowledgements

Peter Nabende is funded through a second NPT Uganda Project (NPT-UGA-238). Thanks to the anonymous reviewers for their valuable comments. Finally, thanks to Erik Tjong Kim Sang for proof reading this manuscript and providing additional comments to enable a much better presentation of the work.

References

Bilmes, J. and G. Zweig (2002), The graphical models toolkit: An open source system for speech and time-series processing, *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 3916–3919.

Filali, K. and J. Bilmes (2005), A dynamic bayesian framework to model context and memory in edit distance learning: An application to pronunciation classification, *Proceedings of the Association for Computational Linguistics (ACL)*, Ann-Arbor, Michigan, pp. 338–345.

Jung, S.Y., S.L. Hong, and E. Paek (2000), An english to korean transliteration model of extended markov window, *Proceedings of the 18th Conference on Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 383–389.

Knight, K. and J. Graehl (1998), Machine transliteration, *Computational Linguistics* **24**, pp. 599–612, Association for Computational Linguistics.

Kondrak, G. and T. Sherif (2006), Evaluation of several phonetic similarity algorithms on the task of cognate identification, *LD'06: Proceedings of the Workshop on Linguistic Distances*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 43–50.

Kumaran, A., M. Khapra, and H. Li (2010), Whitepaper on news 2010 shared task on transliteration mining, Contains information for use in the 2nd shared task on Transliteration Mining.

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

Li, H., V. Pervouchine, and M. Zhang (2009), Report of news 2009 machine transliteration shared task, *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS,2009)*, Association for Computational Linguistics, Suntec, Singapore, pp. 1–18.

Mackay, W. and G. Kondrak (2005), Computing word similarity and identifying cognates with pair hidden markov models, *Proceedings of the ninth Conference on Computational Natural Language Learning (CoNLL 2005)*, Ann-Arbor, Michigan, pp. 40–47.

Murphy, K.P. (2002), Dynamic bayesian networks: Representation, inference, and learning.

Nabende, P., J. Tiedemann, and J. Nerbonne (2010), Pair hidden markov model for named entity matching, *in* Tarek, S., editor, *Innovations and Advances in Computer Sciences and Engineering*, Springer Netherlands, pp. 497–502.

Wieling, M., T. Leinonen, and J. Nerbonne (2007), Inducing sound segment differences using pair hidden markov models, *SigMorphon '07: Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 48–56.

Zweig, G. and S. Russell (1998), Speech recognition with dynamic bayesian networks, *Proc. AAAI-98*, AAAI Press, Madison, Wisconsin.