**University of Alberta**

# Grapheme-to-phoneme conversion and its application to transliteration

by

Sittichai Jiampojamarn

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science


Spring 2011
Edmonton, Alberta

## Examining Committee

Grzegorz Kondrak, Computing Science

Randy Goebel, Computing Science

Dale Schuurmans, Computing Science

Harald Baayen, Linguistics

Anoop Sarkar, School of Computing Science, Simon Fraser University

# Abstract

Grapheme-to-phoneme conversion (G2P) is the task of converting a word, represented by a sequence of graphemes, to its pronunciation, represented by a sequence of phonemes. The G2P task plays a crucial role in speech synthesis systems, and is an important part of other applications, including spelling correction and speech-to-speech machine translation. G2P conversion is a complex task, for which a number of diverse solutions have been proposed. In general, the problem is challenging because the source string does not unambiguously specify the target representation. In addition, the training data include only example word pairs without the structural information of subword alignments.

In this thesis, I introduce several novel approaches for G2P conversion. My contributions can be categorized into (1) new alignment models and (2) new output generation models. With respect to alignment models, I present techniques including many-to-many alignment, phonetic-based alignment, alignment by integer linear programing and alignment-by-aggregation. Many-to-many alignment is designed to replace the one-to-one alignment that has been used almost exclusively in the past. The new many-to-many alignments are more precise and accurate in expressing grapheme-phoneme relationships. The other proposed alignment approaches attempt to advance the training method beyond the use of Expectation-Maximization (EM). With respect to generation models, I first describe a framework for integrating many-to-many alignments and language models for grapheme classification. I then propose joint processing for G2P using online discriminative training. I integrate a generative joint $n$-gram model into the discriminative framework. Finally, I apply the proposed G2P systems to name transliteration generation and mining tasks. Experiments show that the proposed system achieves state-of-the-art performance in both the G2P and name transliteration tasks.

# Acknowledgements

It is my pleasure to thank those who made this thesis possible.

I am deeply grateful to my supervisor, Greg Kondrak, for excellent guidance throughout my research and study at the University of Alberta. His inspiration and enthusiasm help me to explore research in this thesis. Without his guidance, it would not have been possible to publish paper publications used in this thesis. Thank you to my thesis committee members: Randy Goebel, Dale Schuurmans, Harald Baayen and Anoop Sarkar for their contributions to this thesis at the defense. Thank you to Dekang Lin for his comments and suggestions during my candidacy exam. Thank you to Shane Bergsma and Aditya Bhargava for their great work on proofreading this thesis.

Thank you to all NLP members at the University of Alberta and brilliant co-authors contributing in my thesis publications: Tarek Sherif for the many-to-many alignment paper, Colin Cherry for the discriminative training papers, Qing Dou and Shane for stress prediction paper, and Kenneth Dwyer, Shane, Qing, Aditya and Mi-Young Kim for the transliteration papers. Thank you to Ken for his contribution to the name “DirecTL”.

I would like to thank you to Nick Cercone and Vlado Keselj who advised me during my master program at Dalhousie University and continually provided me recommendation letters for scholarship applications and others.

I acknowledge supports from the Alberta Ingenuity Fund and the Alberta Informatics Circle of Research Excellence which are now part of the Alberta Innovates organization.

Lastly but the most importantly, I would like to thank my wife and family for their love, encouragement, and understanding.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The objective of grapheme-to-phoneme conversion (G2P) is to generate phonemes that correspond to a given written word. Phonemes are abstract psychological representations of how words are pronounced in natural speech, while graphemes are representations of words in written language. The G2P task plays a crucial role in speech synthesis systems [Schroeter et al., 2002], and is an important part of other applications, including spelling correction [Toutanova and Moore, 2001], speech recognition [Goel et al., 2010] and speech-to-speech machine translation [Engelbrecht and Schultz, 2005].

In general, G2P can be viewed as a string transduction problem where a system is trained to transform input strings to output strings. Formally, the G2P task can be described as follows: given an input word $s$ containing $n$ graphemes, $s_1 \dots s_n$, the task is to find the $t_1 \dots t_m$ phoneme sequence that corresponds to the input word $s$. For example, the phonemes for the word *abode* are [ə b o d]. Generally, the G2P task requires an alignment algorithm to generate alignments between graphemes and phonemes in the training data. The aligned data provide more precise information to a phoneme generation model. For instance, the alignments for the example above are:

| $a$ | $b$ | $o$ | $d$ | $e$ |
|---|---|---|---|---|
| \| | \| | \| | \| | \| |
| ə | b | o | d | _ |

The "_" phoneme, called the *"null phoneme"*, is added to represent the silent sound produced by the grapheme "e" in the example. In many cases, the null phoneme is also added for making one-to-one alignments possible.

Converting a word into its phoneme representation is a non-trivial task. Dictionary-based approaches cannot achieve this goal reliably due to unseen words and proper names. Furthermore, the construction of even a modestly-sized pronunciation dictionary requires substantial human effort for each new language. Effective rule-based approaches can be de-

signed for some languages such as Spanish; however, Kominek and Black [2006] show that in languages with a less transparent relationship between spelling and pronunciation, such as English, Dutch, or German, the number of letter-to-sound rules grows almost linearly with the lexicon size. Therefore, most recent work in this area has focused on machine learning approaches.

Many data-driven techniques have been proposed for grapheme-to-phoneme conversion systems, including neural networks [Sejnowski and Rosenberg, 1987], hidden Markov models [Taylor, 2005], instance-based learning [Bosch and Daelemans, 1998, Daelemans and Bosch, 1997], constraint satisfaction [Bosch and Canisius, 2006], and decision trees [Black et al., 1998]. Data-driven approaches to G2P generally require aligned training data of graphemes and phonemes. A one-to-one alignment assumption is typically assumed for simplicity. Phoneme generation models are then trained using the alignments, viewing the task as either a classification or a sequence prediction problem.

I aim to develop an automatic G2P system that learns from available word-phoneme examples and outperforms all other existing systems. Before applying any learning techniques to this problem, there are three issues one should consider. First, input words and output phonemes are embedded with some hidden structures among the two spaces. These hidden structures are called **"alignments"**. Discovering these alignments is required in order to train G2P systems using many learning techniques. Second, words and phonemes are naturally structured sequences, so that classification-based techniques are not able to capture the output structure information. However, they are good at representing grapheme contexts as input features and generating each phoneme subsequence output. Third, each phoneme is influenced by both the hypothetical grapheme(s) that generates it (as known by the links in the alignment data) and the grapheme context surrounding the grapheme. For example, in the word-phoneme *"abode"* [ə b o d], the phoneme [o] is not only generated by the grapheme "o" but is also influenced by other graphemes surrounding it, like the graphemes "d" and "e". With the possibility of long dependencies in the grapheme sequence affecting the generation of an output phoneme, sequence-based learning techniques such as generative hidden Markov models (HMMs) are a poor fit for this problem. Discovering a learning technique that better fits this task is challenging.

The similar task of name transliteration is one in which, given a name written in a source language, we seek a phonetically equivalent name written in a target language. The transliteration task is another instance of a string transduction problem, and it is similar to G2P in many aspects. In fact, the name transliteration system of Knight and Graehl

[1998] includes G2P components. The idea is to convert source language graphemes to their corresponding phonemes. The system then learns mapping relationships between source language phonemes and target language phonemes before converting the target language phonemes to their corresponding graphemes, written in the target script. This approach therefore consists of a G2P component in the source language and the reverse process, P2G, in the target language. The name transliteration system of Li et al. [2004] present a direct-conversion-based approach that generates target language names from the source language without intermediate phoneme representations. This approach is conceptually similar to the well known G2P joint $n$-gram model proposed by Bisani and Ney [2002]. This suggests a strong relationship between G2P and name transliteration tasks in general.

The Named Entity Workshop (NEWS) shared tasks on name transliteration [Kumaran et al., 2010, Li et al., 2009, 2010] are interested in the development of language-independent name transliteration systems. In the transliteration generation tasks, the available training data are similar similar to the available training data for G2P; the data consist of transliteration names written in source and target languages. Like G2P, there is no alignment information in the training data indicating which substring of a target language name is a production of which substring of a source language name. To train a system, we need the training data to be aligned before starting the learning process. Unlike standard evaluation of G2P, there are multiple correct or accepted transliteration targets. Training a system with multiple correct answers in the training data is challenging.

Like in many other supervised natural language processing applications, the quality of a transliteration system largely depends on the size and quality of transliteration example pairs. Developing large corpora is both time consuming and expensive. Transliteration mining in the NEWS shared tasks [Kumaran et al., 2010] is interested in extracting transliterated names from parallel text. I aim to develop a system that is trained on a small list of transliterated names and is able to identify name transliteration pairs in the text, written in source and target languages.

In this thesis, my overall research objective is to develop a G2P system that improves over existing systems by employing novel advanced techniques in both alignment and generation. I apply such approaches to not only the G2P conversion tasks but also to name transliteration generation and mining tasks.

## 1.1 Contributions

In this thesis, my research contributions are focused on G2P conversion and name transliteration. For G2P, I present a many-to-many alignment approach, known as the **"m2m-aligner"** [Jiampojamarn et al., 2007]. As mentioned previously, machine learning techniques require the training data to be aligned before starting the learning process. Previous work generally assumed one-to-one alignments for simplicity. This assumption limits one grapheme to be aligned with one phoneme in the output. Although these alignments make the original G2P problem simpler and more suitable to many multi-class classification techniques, the alignments suffer from two fundamental problems: (1) double graphemes and (2) double phonemes, which are discussed in detail in Section 2.1.1. My many-to-many alignment approach is proposed to fix these problems. The alignment results are more accurate from both human and machine learning perspectives. I propose alternative alignment methods including phoneme-based, integer linear programming-based, and alignment-by-aggregation approaches [Jiampojamarn and Kondrak, 2010]. I conduct an in-depth study to demonstrate the close relationship between alignment quality and G2P conversion performance.

I first propose a pipeline framework for applying the many-to-many alignments and a phonetic language model to G2P classifiers [Jiampojamarn et al., 2007]. This approach combines the benefits of hidden Markov models (HMMs) with the flexibility of supervised classification techniques. Naturally, classification techniques are unaware of sequential output structures but they easily make use of wide grapheme context to individually predict each sub-phoneme in the sequence. HMMs are sequence-based techniques that generate output sequences based on the transition and emission probabilities. The output structure information is encoded in the transition probability, and source-side information is encoded in the emission probability. While the G2P systems based solely on HMMs tend to perform poorly, I show that using an HMM-like model to correct the output phoneme sequence generated by a local classifier yields an improvement in prediction accuracy.

Later, I collapse the pipeline framework and unify it with a joint processing framework based on online discriminative training. This novel framework is known as DirecTL [Jiampojamarn et al., 2008]. The discriminative training approach has the ability to incorporate a rich number of features. Phoneme sequence information is easily represented as features in the learning method. A phrase-based decoder joins together the phoneme gen-

eration module and the grapheme segmentation module based on the many-to-many alignments. The joint processing framework optimizes the learning parameters for G2P as a single model, solving the error propagation problem in the pipeline framework. I further conduct a study to evaluate this system while incorporating a state-of-the-art phonetic stress assignment system [Dou et al., 2009]. The proposed system outperforms the well-known speech synthesis system Festival[1]. One observation with the joint technique, however, is that the model must commit to imperfect alignments produced by the m2m-aligner. I investigate this potential issue by applying a training approach without explicit alignments, based on the method of Liang et al. [2006]. The training process is similar to a coordinate descent algorithm that is applied to Latent Support Vector Machines (LSVMs) [Felzenszwalb et al., 2008]. Finally, I combine a joint $n$-gram model [Bisani and Ney, 2002, 2008] with the DirecTL system; the combined system is called "DirecTL+" [Jiampojamarn et al., 2010a]. I conduct experiments that evaluate the proposed system on the same data splits as used in [Bisani and Ney, 2008]. DirecTL+ outperforms both the original DirecTL and the joint $n$-gram approach, establishing a new standard in state-of-the-art performance for G2P conversion.

Furthermore, I apply the DirecTL framework to name transliteration generation and mining and evaluate using the data provided by the NEWS 2009 and 2010 Machine Transliteration Shared Tasks [Kumaran et al., 2010, Li et al., 2009, 2010]. I present a training approach that incorporates multiple outputs per input for the name transliteration generation task. This approach enables the DirecTL framework to train with multiple valid correct answers per input name. The results [Jiampojamarn et al., 2009, 2010b] illustrate DirecTL's proficiency for the name transliteration generation task. It does not require any specific language knowledge in order to achieve state-of-the-art performance on several language pairs. However, language-specific approaches for Chinese and Korean are also tested as pre-processing steps to provide greater information for the corresponding writing systems. The pre-processing steps for these languages help the system to generate better alignments and better output sequences.

For name transliteration mining, I present classification-based and generation-based approaches that are based on the m2m-aligner and DirecTL systems [Jiampojamarn et al., 2010b]. To make a decision on whether a candidate pair is a transliteration pair, I present an approach that uses the DirecTL system to generate transliteration words and I compare the generated words with the words in the candidate list via a string similarity function. The

[1] `http://www.cstr.ed.ac.uk/projects/festival`

successful results achieved in the shared tasks confirm the effectiveness of this approach.

In summary, I present alignment and phoneme generation techniques for G2P conversion tasks. The improvements achieved by the proposed methods are based on observations made in existing systems. Then, I apply the proposed methods to name transliteration generation and mining tasks. I illustrate that the proposed approaches are efficient for both the G2P and the transliteration tasks. I establish the state-of-the-art in performance on both tasks.

## 1.2 Outline

The structure of this thesis is as follows. First, in Chapter 2, I describe grapheme-to-phoneme work in the literature, including grapheme-to-phoneme one-to-one alignment methods and phoneme generation models based on classification and sequence prediction techniques. Then, in Chapter 3, I introduce the many-to-many alignment approach as well as phonetic alignment, integer linear programming-based alignment, and alignment-by-aggregation. In Chapter 4, I describe a grapheme chunking model and language model for classification-based G2P approaches. Then, the joint processing framework and online discriminative training methods (DirecTL and DirecTL+) are presented and evaluated. I present the DirecTL framework for name transliteration generation and mining tasks in Chapter 5. I provide the conclusion of this thesis in Chapter 6.

# Chapter 2

# Related Work

Grapheme-to-phoneme (G2P) conversion[1] has a long history in the speech synthesis area. Inspired by Chomsky and Halle [1968], the conversion of graphemes to phonemes is possible if there is sufficient contextual information around the graphemes [Damper et al., 1999]. The Chomsky and Halle [1968] work inspired the early rule-based system of [Elovitz et al., 1976] which was proposed as an automatic rule-based system for English. It contains 329 phonological rules that are in the format of "$A[B]C \rightarrow D$"; where $A$ and $C$ are the left and right context graphemes and $D$ is the phoneme(s) corresponding to the grapheme $B$. Over time, machine learning techniques have been developed; modern G2P systems do not rely on handcrafted rules but rather use available word-phoneme example pairs for training.

However, the inspiration of Chomsky and Halle [1968] still plays a crucial role as grapheme context features in classification-based approaches [Black et al., 1998, Bosch and Canisius, 2006, Bosch and Daelemans, 1998, Daelemans and Bosch, 1997, Kienappel and Kneser, 2001, Sejnowski and Rosenberg, 1987, Suontausta and Tian, 2003], or grapheme substrings in generative models [Bisani and Ney, 2002, 2008, Chen, 2003, Damper and Eastmond, 1997, Marchand and Damper, 2000, Taylor, 2005].

There are two key components in both classification and generative systems that allow training from word-phoneme example pairs. The first component is to discover hidden structures between graphemes and phonemes, called *alignments*. Alignments essentially allow the G2P system to learn what phoneme to generate for each grapheme and its context, allowing us to attack the problem sequentially grapheme by grapheme. The alignments are provided to the phoneme generation models either explicitly, having a separate model to generate the alignments, or implicitly, integrated within the phoneme generative models [Bisani and Ney, 2002, 2008, Chen, 2003, Taylor, 2005]. The second component is a

[1] also known as letter-to-phoneme (L2P) or letter-to-sound (L2S) conversion

learning mechanism to train a model to generate output phonemes given words. There are two paradigms for training:

1. train as a classification problem where each grapheme in a word generates a phoneme or phonemes without knowledge of previously generated phonemes in the same word.

2. train as a sequence modeling or tagging problem which takes previous decisions in to consideration for the current decision.

The performance of G2P systems are reported in word accuracy, and phoneme accuracy. The word accuracy is calculated by counting the number of fully correct phoneme sequences given testing words. The phoneme accuracy is calculated using either the Hamming distance or Levenshtein distance between the gold-standard outputs and the generated sequences to find the number of correct phonemes.

Generally, training data for G2P conversion systems are available in the form of word-phoneme pairs with no explicit information indicating individual grapheme-to-phoneme relationships. While these relationships are *"hidden"* in the training data, humans naturally have an intuition of grapheme-to-phoneme *"alignments"* given a word-phoneme pair. For example, the word *"accuse"* [ @ k j u z ] has grapheme-to-phoneme alignments shown in Figure 2.1. The "_" represents a special null phoneme that indicates a silent sound for grapheme *"e"* in the word.


Figure 2.1: Alignment example for the word *"accuse"* with its phoneme output.

To simplify the conversion task, these grapheme-phoneme alignments must be discovered so that phoneme generation models can infer the relationship between each/substring of graphemes in the input word and each substring of phonemes in the output phonemes. In very early work, graphemes and phonemes were aligned by hand before a grapheme-phoneme prediction model was trained [Sejnowski and Rosenberg, 1987]. In later work, the alignment step was performed in an automatic fashion using an expectation maximization (EM) based algorithm [Dempster et al., 1977] with no (or minimal) human interaction. These methods are based on a one-to-one alignment assumption for simplicity [Black et al., 1998, Daelemans and Bosch, 1997, Damper et al., 2005].

## 2.1 One-to-one EM alignment

The one-to-one assumption assumes that each grapheme can be aligned to one phoneme or the null phoneme which represents a silent sound. A general framework is presented in Algorithm 1. The process starts with the initial probability of mapping a grapheme $s$ to a phoneme $t$, $P(s,t)$ in line 1 and iteratively re-computes the probability table based on the new alignments found under the current $P(s,t)$, in line 4 and 5, until the probability table converges. Finally, the $P(s,t)$ is returned to produce the alignments in the training data. Note that while Black et al. [1998] compute $P(s,t)$ probabilistically, Daelemans and Bosch [1997] and Damper et al. [2005] treat $P(s,t)$ as raw counts or scores of mapping grapheme $s$ to phoneme $t$. I now explain each of these steps in detail.

Black et al. [1998] proposed (1) the hand-seeded method and (2) the epsilon scattering method to initialize $P(s,t)$. The hand-seeded method starts with an explicit list of which phonemes (or multiple phonemes) each grapheme can be mapped to, and finds the best possible alignments for each word-phoneme pair in the training data. The initial probability $P(s,t)$ is calculated based on the mapping counts. While the hand-seeded method requires human effort to produce the allowance list, the epsilon scattering method obtains the counts without the list. The initial probability table in this case starts by mapping all possible alignments in the one-to-one fashion between graphemes and phonemes by introducing possible null phoneme positions. For example, *abode* [ ə b o d ] has five possible positions where a null phoneme can make an alignment. Instead of scattering null phoneme positions, Daelemans and Bosch [1997] proposed the shifting method to count with different scores. The shifting is repeated at most 3 times, so that the possible counting scores are 8, 4, 2 and 1. For example, *rookie* [ r u k i ] has three possible alignments: (1) no shift *rookie* [ r u k i _ _ ], (2) one right shift [ _ r u k i _ ] and (3) two right shifts [ _ _ r u k i ]. Each mapping takes 8 counts for no shift, 4 counts for one right shift and 2 for two right shifts cases. Damper et al. [2005] proposed a simple way to obtain the initial $P(s,t)$ by counting appearances between a grapheme $s$ and phoneme $t$ in the same word regardless of positions.

The next step in the algorithm is to find new alignment paths based on the current $P(s,t)$ (line 4). The maximum likelihood path can be found by using standard dynamic programming. Let $C(i,j)$ be the score entry at grapheme position $i$ and phoneme position $j$, with initially $C(0,0) = 0$. The recursive equation for dynamic programming is shown in Equation 2.1. The alignment path can be reconstructed by back-tracking from the maximum score $C(I,J)$; where $I$ and $J$ are the numbers of graphemes and phonemes in the word,

**Algorithm 1** One-to-one EM alignment training.

**Input:** word-phoneme training examples
**Output:** mapping probability table $P(s,t)$
1: Initialize probability $P(s,t)$.
2: **for** $K$ iterations over the training set **do**
3: **for all** word-phoneme pair in the training set **do**
4: find alignments path based on $P(s,t)$.
5: re-compute $P(s,t)$ based on the new alignments found.
6: **return** $P(s,t)$

respectively.

$$C(i,j) = \max \begin{cases} C_{i-1,j-1} + P(s_i, t_j) \\ C_{i-1,j} + P(s_i, \_) \\ C_{i,j-1} + P(\_, t_j) \end{cases} \tag{2.1}$$

Allowing the mapping between a null grapheme and phoneme is problematic during the phoneme generation phase in which the null grapheme does not exist in words. To disallow the null grapheme mapping, $P(\_, s_j)$ can be set to a large negative number. This prevents the decoder from mapping a phoneme with a null grapheme as in [Black et al., 1998, Daelemans and Bosch, 1997]. Similarly, $P(s_i, \_)$ can be set to a constant cost for mapping null phonemes [Damper et al., 2005].

Finally, for each training iteration the new $P(s,t)$ is re-computed based on the new alignment found as in line 5. The training process stops when $P(s,t)$ converges.

### 2.1.1 Discussion on one-to-one EM alignment

Although alignments obtained from one learning method are not always identical to the human-generated alignments or to alignments from other aligners, they are sufficient to provide useful information to a phoneme generation model. These aligners can be trained in an unsupervised manner with no or minimal human supervision. In most cases, they correctly capture those silent sound cases (e.g. *abode* [ ə b o d ]). The one-to-one mapping assumption keeps the computation simple; however, there are two problems:

1. Double graphemes: when two graphemes map to one phoneme (e.g. *sh* → [ ʃ ], *ph* → [ f ])

2. Double phonemes: when one grapheme maps to two phonemes (e.g. *x* → [ k s ], *u* → [ j u ])

First, the double grapheme problem occurs when two graphemes map to one phoneme resulting in a shorter phoneme string. For example, in the word *king* [ k ɪ ŋ ], the graphemes

*ng* intuitively produce the phoneme [ ŋ ] together. To produce one-to-one alignments, the null phoneme has to be aligned with either the grapheme *n* or *g*, neither of which is intuitively the correct alignment. These incorrect alignments can potentially cause the phoneme prediction model to produce null phonemes for either *n* or *g* grapheme.

Second, the double phoneme problem arises in those cases where one grapheme produces two phonemes. For example, in the word *fume* [ f j u m ], the vowel *u* generates both [ j ] and [ u ] phonemes. One possible alignment path is to add a null grapheme in the word string, and to align the null grapheme with either [ j ] or [ u ] phoneme. Adding a null grapheme not only results in incorrect alignments, which confuse the phoneme generation model, but also lead to another problem: where should the null grapheme be added in the word string during generation phase, since it does not exist in the orthographic side? Another possible solution for the double phoneme problem is to create a new phoneme by merging phonemes [ j ] and [ u ]. This solution requires an expert to construct a new phoneme list (e.g. the handed-seed method [Black et al., 1998]). Figure 2.2 shows an one-to-one alignment of the example word *accuse* [ @ k u z ]. The alignment is possible by merging both [ j ] and [ u ] phonemes to a new phoneme [ U ] and aligning the second grapheme *c* with a null phoneme.

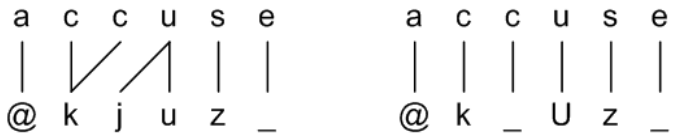


Figure 2.2: One-to-one and many-to-many alignment examples for the word *"accuse"*.

These two fundamental problems exist in the one-to-one alignments. The one-to-one assumption thus makes the task simple but it limits the ability to train a phoneme generation model from the alignments.

The many-to-many alignment method was proposed for the grapheme-to-phoneme conversion task by Jiampojamarn et al. [2007]. It relaxes the one-to-one assumption allowing multiple graphemes to align with multiple phonemes. The method requires no handcrafted list and learns in an unsupervised manner without pre-aligned examples.

## 2.2 Grapheme-to-phoneme conversion

Once the alignments are discovered in the training data, we can use them to explicitly express grapheme-to-phoneme productions. G2P can then be viewed either as a multi-class

classification problem, where each sub-phoneme output is drawn directly from the focused grapheme and its context (surrounding graphemes) without considering the phoneme sequence output, or as a sequence prediction problem, which takes into account the grapheme sequence input and phoneme sequence output.

### 2.2.1 Classification-based approaches

In classification-based approaches, each phoneme is predicted independently using a classifier such as a neural network [Sejnowski and Rosenberg, 1987], instance-based learning [Bosch and Daelemans, 1998, Daelemans and Bosch, 1997] and decision tree [Black et al., 1998, Kienappel and Kneser, 2001, Suontausta and Tian, 2003]. These classifiers predict a phoneme for each input grapheme using the grapheme, called the *"focus grapheme"*, and its context graphemes as features. The focus grapheme is the most important feature in the prediction, while the farther the context grapheme is, the less information it contributes to the classifiers. With the same distance, right context graphemes are slightly more important than left context graphemes [Daelemans and Bosch, 1997]. In general, these methods use 3 to 5 graphemes both before and after the focus grapheme as context features, depending on languages and data sets. For English, reported in NETtalk [Sejnowski and Rosenberg, 1987], a performance improvement was found with increasing the context window size from 3 to 5. These classification-based methods leverage the structure of the input, encoded in the features, by using context grapheme information, but they ignore the phoneme structure in the output.

NETtalk is one of the first grapheme-to-phoneme systems, and a well-known neural net application. The method is based on the error back-propagation neural net training. The focus grapheme and its context graphemes are encoded as binary nodes for predicting a phoneme output. Each input grapheme is encoded with 29 nodes: 26 nodes for the English alphabet and 3 nodes for punctuation marks. The network consists of one hidden layer of 80 hidden units. The output phonemes are encoded with 26 binary nodes representing 21 articulatory features (voiced, velar, stop, and so on) and 5 stress and syllable boundary features (strong, weak, left, and so on).

Daelemans and Bosch [1997] proposed to use IG-Tree, a variation of instance-based learning, for the grapheme-to-phoneme problem. The IG-Tree method uses information gain to grow the decision tree that stores grapheme-phoneme examples. The IG-Tree and decision tree methods are different in the sense that IG-Tree uses the information gain to evaluate features once whereas the decision tree method re-evaluates features at each grow-

ing tree step. While the decision tree method usually includes pruning to avoid over-fitting, the IG-tree is constructed until all the training phoneme set ambiguity is resolved, without pruning the tree. Interestingly, Black et al. [1998] reported that growing the decision tree with an early stop criteria actually decreases the performance.

A similar approach to the IG-tree was proposed by Bosch and Daelemans [1998]. It is based on memory-based learning IB1-IG. Instead of walking in the IG-tree, the method finds the most similar training instance to predict the new testing instance. The similarity function is based on the Euclidean distance with each feature weighted by its information gain value.

### 2.2.2 Sequence-based approaches

A natural way to view grapheme-to-phoneme conversion is to consider it as a sequence modeling or tagging task. An input word $X$ consists of a sequence of graphemes $x_1 \dots x_n$ and its corresponding output $Y$ is a sequence of phonemes $y_1 \dots y_m$. The major difference between classification-based and sequence-based approaches is that the latter considers previous phoneme decisions in order to predict the current sub-output phoneme.

Taylor [2005] describes applying a hidden Markov model (HMM) to the G2P task. The method formulates grapheme sequences as observation states and phoneme sequences as hidden states. The phoneme sequence output is the most probable sequence based on the transition and emission probabilities, using the following equation; where $X$ is a sequence of graphemes (input word) and $Y$ is a sequence of phonemes:

$$\hat{Y} = \arg\max_{Y} P(X|Y)P(Y) \tag{2.2}$$

The HMM framework is based on Baum-Welch training. It allows one phoneme state to generate up to four graphemes. This allowance in the decoder avoids introducing the null phoneme to the problem. The approach does not require the training data to be aligned separately but optimizes both alignment process in the decoder and phoneme generation parameters within one HMM framework. However, the performance achieved by the HMM framework is far worse than classification-based approaches, even with some ad-hoc preprocessing fixes, which tune the framework to correct errors. These inferior results are mainly caused by the fact that the HMM framework lacks the capability of using the grapheme context information directly. These features have been shown to be important in classification-based approaches.

Joint n-gram models [Bisani and Ney, 2002, 2008, Chen, 2003] achieve good G2P results by training the models on grapheme-phoneme substring pairs, so that sequence information in both the grapheme and phoneme sides directly contribute to the models. Bisani and Ney [2002] incrementally create grapheme-phoneme chunks of sizes ranging from 1 to 6. They reported an optimal size of 2 in both graphemes and phonemes for English and German data sets. Given a word $X$ and its corresponding phoneme $Y$, the method then estimates the joint probability $P(X, Y)$ from grapheme-phoneme segmentations called "chunks" $c_1 \dots c_L$ as in Equations 2.3 and 2.4; where $S(x, y)$ is the set of all possible joint segmentations of word $X$ and phoneme sequence $Y$. The maximum likelihood training is based on the EM algorithm. The most likely phoneme sequence $Y^t$ given a new test word $X^t$ can be found by $\arg\max_{Y^t} P(X^t, Y^t)$.

$$P(X, Y) = \sum_{C \in S(x,y)} p(C = c_1, \dots c_L) \tag{2.3}$$

$$P(C = c_1 \dots c_L) = \prod_{i=1}^{L} P(c_i | c_1 \dots c_{i-1}) \tag{2.4}$$

Pronunciation by Analogy (PbA) [Damper and Eastmond, 1997, Marchand and Damper, 2000] also considers substring graphemes and phonemes for the G2P task. The method produces a phoneme sequence output for an input word $X$ by finding the least number of segmented grapheme sequences in the training examples that can form the word $X$ with the highest score of the phoneme sequence path. The grapheme-phoneme chunks are based on the longest common subsequence between each training example and a new word. It requires alignments to be drawn in the training data which can be trained by using the one-to-one aligners described in Section 2.1. To illustrate, consider Figure 2.3; each arc represents the phoneme substring corresponding to the input grapheme substring. The number on the arc represents the number of occurrences of the grapheme-phoneme substrings in the training example. For example, the substring *long* [l a n J] appears twice, while *lon* [l a n] and *lon* [l o n] appear twice and once, respectively, in the training examples. PbA is a lazy learning method that stores all training examples and creates the path during the transcription step. It takes the shortest path as the output. Moving from one node to another node in the graph is counted as a path with length 1. If there are more than one candidates with the same length (e.g. there are 6 candidates with length 3 in the example), PbA selects the path which has the highest number of occurrences summed along the arcs [Damper and Eastmond, 1997] or uses 5 scoring strategies to rank the candidates [Marchand and Damper,

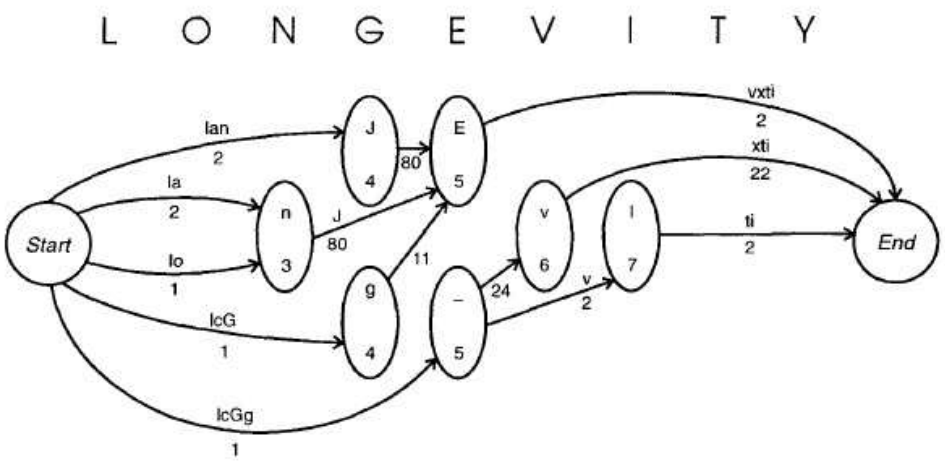


Figure 2.3: Example pronunciation for the word longevity showing only the arcs contributing to the phoneme sequence output. The figure is taken directly from [Marchand and Damper, 2000].

2000].

The constraint satisfaction inference (CSInf) approach [Bosch and Canisius, 2006] improves the performance of the classification-based approach [Bosch and Daelemans, 1998] by predicting, for each grapheme, a trigram of phonemes consisting of the previous, current, and next phonemes. The core learning technique is based on instance-based learning; it takes the same grapheme context features as in the standard classification-based approach. By predicting a trigram output, CSInf increases the number of classes from the original problem by polynomial order 3. The final output is based on the overlapping output class information by taking the output sequence that satisfies the most unigram, bigram, and trigram agreement constraints. The overlapping predictions improve G2P performance mainly by repairing imperfect one-to-one alignments.

## 2.3 Learning for structured outputs

Grapheme-to-phoneme learning is closely related to structured learning techniques including hidden Markov models (HMMs) [Rabiner, 1989], averaged perceptron algorithm [Collins, 2002], Support Vector Machines for structured outputs (SVM$^{struct}$) [Tsochantaridis et al., 2004], and conditional random fields (CRFs) [Lafferty et al., 2001]. As reported in [Taylor, 2005] and discussed in Section 2.2.2, HMMs are a poor fit to the G2P task because they lack the capability to utilize contextual information in the input grapheme

sequences. The HMM model assumes that each phoneme sub-output $y_i$ only depends on its observation $x_i$ and its preceding phonemes $y_{i-1} \dots y_{i-M}$, where $M$ is the Markov order assumption. The model uses no direct information of contextual graphemes $x_{i-C} \dots x_{i+C}$, where $C$ is an allowing window context size.

Collins [2002] proposed a discriminative averaged perceptron algorithm that extends the generative HMMs. As it is discriminative, the averaged perceptron model works with a notion of "score" instead of probability as in HMMs. The phoneme sequence output is the maximum linear combination of scores for each sub-output $y_i$ given the grapheme sequence input $x$ shown in Equation 2.5; where $w$ represents feature weight parameters, $\Phi(x, y_i)$ is the feature vector indicating evidence found between $x$ and $y_i$, and $I$ is the length of the input sequence. The $GEN(x)$ function indicates all possible output sequences $y$ that can be generated by the input string $x$. If the model assumes the one-to-one constraint, the $\arg\max$ operation is the Viterbi algorithm. Otherwise, a segmentation process is required over the input sequence to create $I$ input units. Alternatively, a phrase-based decoder [Zens and Ney, 2004] can be applied instead of the Viterbi decoder to find the most likely grapheme units as well as the output sequence.

$$\hat{Y} = \arg \max_{y=y_{1\dots I} \in GEN(x)} \sum_{i=1}^{I} w \cdot \Phi(x, y_i) \tag{2.5}$$

Since the model is based on the summed score, it provides the ability to observe evidence more freely with no limitation on transition and emission probabilities as in HMMs. In fact, the transitions and emissions can be a subset of $\Phi(x, y_i)$ expressing the focus grapheme $x_i$ and preceding phonemes $y_{i-1} \dots y_{i-M}$. Unlike HMMs, $\Phi(x, y_i)$ typically uses indicator values that represent present or absent evidence. The weight vector $w$ assigns how much each component of $\Phi$ contributes to the total score. The training process to find the optimal $w$ is an online learning method that iterates through the training data. For each example, the model finds the most likely output $\hat{y}$ based on the current $w$. The weight vector is then updated such that it prefers the correct $y$ over the incorrect $\hat{y}$, as shown in Equation 2.6.

$$w \leftarrow w + \Phi(x, y) - \Phi(x, \hat{y}) \tag{2.6}$$

The model trains until $w$ converges. The average of all weight vectors that are seen during training is used in the final model instead of instead of the final weight vector, providing a better generalized model [Freund and Schapire, 1999].

Support Vector Machines for structured outputs[2] (SVM$^{struct}$) [Tsochantaridis et al., 2004] find the optimal weight vector $w$ by formulating the problem as the following quadratic program:

$$\min_{w,\xi} \frac{1}{2} \parallel w \parallel^2 + \frac{C}{n} \sum_{k=1}^{n} \xi_k, \quad \text{s.t. } \forall k, \xi_k \geq 0 \tag{2.7a}$$

$$\forall k, \forall y \in \mathcal{Y} - y_k : w \cdot [\Phi(x_k, y_k) - \Phi(x_k, y)] \geq 1 - \xi_k \tag{2.7b}$$

The objective function in Equation 2.7a ensures a unique solution where the norm of the weight vector is minimized. The summation of slack variables $\xi_k$ allows training errors, called the "soft-margin SVM"; $C$ controls the trade-off between training error minimization and margin maximization. Equation 2.7b is a set of constraints to ensure that the score of the correct output sequence for the $k^{th}$ training instance, $y_k$, is larger than other incorrect sequences $y$ by at least $1-\xi_k$. The right hand side of the inequality is based on the zero-one loss function; the loss function describes how wrongly an incorrect output sequence to the correct one (e.g. Hamming distance). To accompany the structured outputs, one can use an arbitrary loss function $\Delta(y_k, y)$ instead of the zero-one loss function. The loss function can be applied to re-scale either the slack variable [Tsochantaridis et al., 2004] by replacing the right hand side of the equation with $1 - \frac{\xi_k}{\Delta(y_k,y)}$ or the margin [Taskar et al., 2004] using $\Delta(y_k, y) - \xi_k$.

Equation 2.7 cannot be solved directly due to the set of constraints. Enumerating all possible output sequences $y$ for each training example is infeasible with an extremely large set of constraints. An iterative training process [Tsochantaridis et al., 2004] is applied to find a small set of constraints that is sufficiently needed for the optimization problem. The algorithm iterates over the training examples until the set of constraints converges.

$$y'_k = \arg \max_{\hat{y_k} \in \mathcal{Y} - y_k} w \cdot [\Phi(x_k, \hat{y_k}) - \Phi(x_k, y_k)] + \Delta(y_k, \hat{y_k}) \tag{2.8}$$

At each iteration, it finds the output $y'_k$ that has the most violated constraint as shown in Equation 2.8 (margin re-scaling) and it adds $y'_k$ to the working set only if the value in the $\arg\max$ is larger than the maximum value in the current working set. Then, $w$ is optimized using the current working set of constraints. Note that at each iteration, there will be only one constraint added into the working set. Tsochantaridis et al. [2004] showed that the proposed algorithm finishes within a polynomial number of iterations and finds a weight vector $w$ that satisfies the constraints in Equation 2.7.

[2] `http://www.cs.cornell.edu/People/tj/svm_light/svm_struct.html`

Conditional random fields (CRFs) [Lafferty et al., 2001] are a specific example of log-linear models that find the optimal weight parameter $\hat{w}$ such that it maximizes the conditional log likelihood $\log P(y|x;\hat{w})$. The optimal weight $\hat{w}$ is found using the following equation:

$$\hat{w} = \arg\max_{w} \sum_{x,y} \left[ \log \left( \frac{\exp(w \cdot \Phi(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(w \cdot \Phi(x,y'))} \right) \right] \tag{2.9}$$

Intuitively, the model ensures that the optimal weight parameter $\hat{w}$ provides a greater conditional probability for the true output sequence $y$ than other incorrect sequences $y'$ for a given input instance $x$. Solving the $\arg\max$ in Equation 2.9 is usually performed via numerical optimization. The simplest approach to this optimization problem is gradient ascent since the equation is strictly concave: any local optimum is guarantee to be the global optimum. In practice, this optimization procedure requires many iterations before it converges [Sutton and McCallum, 2006]. Faster optimization methods, such as BFGS [Bertsekas, 1999] and L-BFGS [Byrd et al., 1994], are generally used instead. Even with an efficient optimization, training a CRF model can be expensive due to the marginal probability requirement for each training example per each gradient computation.

In this thesis, I propose an approach for G2P generation model that combines the benefits of the hidden Markov model and classification techniques. Classification-based techniques for G2P efficiently incorporate wide grapheme context information in order to predict phonemes, while sequence-based approaches take phoneme sequence information into account when they predict each sub-output phoneme. The proposed method here uses a classifier to produce phonemes with classifier confidence values. The confidence values are normalized into values between 0 and 1, then an HMM-like model takes these confidence values as emission probabilities instead of individual grapheme observations.

Later, I propose a sequence-based G2P system that naturally incorporates both wide grapheme context and output sequence information in the model. The proposed method stands between the averaged perceptron model of Collins [2002] and more expensive models such as SVM$^{struct}$ and CRFs in term of computational cost. An online large margin-based update method [Crammer and Singer, 2003] is applied instead of using the simple perceptron rules. Unlike SVM$^{struct}$, the proposed method requires only a small fixed number of constraints to optimize the weight parameter $w$ instead of a number that is polynomial in training size. Unlike CRFs, it requires only the computation of the posterior function without requiring the marginal probability for each optimization step.

# Chapter 3

# Grapheme-to-phoneme alignment

In this chapter, I present and evaluate approaches to grapheme-to-phoneme alignments. As discussed in Section 2.1, I emphasized the underlying problems of the traditional one-to-one grapheme-phoneme alignments. The one-to-one assumption creates two fundamental problems: (1) the double grapheme problem and (2) the double phoneme problem. A direct solution to these fundamental problems is to relax the one-to-one assumption to allow many-to-many alignments. A novel approach for many-to-many alignment, "m2m-aligner", is presented in Section 3.1. This research is joint work with Grzegorz Kondrak and Tarek Sheif, published in [Jiampojamarn et al., 2007]. An implementation of m2m-aligner, written in C++, is publicly available online with source code at, `http://code.google.com/p/m2m-aligner/`, for research, commercial and other purposes.

Most previous alignment methods, including traditional one-to-one alignment methods and the proposed m2m-aligner, are based on the Expectation Maximization (EM) algorithm. To advance the grapheme-phoneme alignment technology, alternative methods are presented in Sections 3.2 and 3.3. This is joint work with Grzegorz Kondrak published, in [Jiampojamarn and Kondrak, 2010]. First, in Section 3.2, the phonetic alignment (ALINE) approach is described. The key idea of this approach is to generate grapheme-phoneme alignments using their phonetic similarity relationship via the International Phonetic Alphabet (IPA). Originally, the ALINE algorithm was proposed by Kondrak [2000]. It aims to create an alignment between two strings of phonemes. Next, the integer linear programming alignment approach is described in Section 3.3.1. It is inspired by the minimal model approach of Ravi and Knight [2009]. The idea is to find the least complex optimal set of possible grapheme-phoneme mappings that describe the word-phoneme examples in the training data set, and to restrict the EM training to draw from the optimal

**Algorithm 2** Many-to-many EM alignment training.

**Input:** word-phoneme training examples, $maxS, maxT$
**Output:** $\delta$

1: Initialize mapping probability table $\delta$
2: **for** $K$ iterations over the training set **do**
3: **for all** word-phoneme pair $(s_1^I, t_1^J)$ in the training set **do**
4: $\gamma \leftarrow$ *Expectation-many2many*$(s_1^I, t_1^J, maxS, maxT, \gamma, \delta)$
5: $\delta \leftarrow$ *Maximization-Step*$(\gamma)$
6: $\gamma \leftarrow 0$
7: **return** $\gamma$

set. This approach prevents the EM training model from introducing rare and unnecessary grapheme-phoneme mappings. Finally, the alignment-by-aggregation approach presented in Section 3.3.2 limits the complexity of the alignment model in its search; it post-processes $n$-best grapheme-phoneme alignments in order to create complete alignment coverage.

The proposed grapheme-phoneme alignment methods are evaluated and presented in Section 3.4. Intrinsic evaluation is performed by comparing the generated alignments to a manually created gold-standard set. The extrinsic evaluation uses two different generation techniques to perform grapheme-to-phoneme conversion on several different data sets.

## 3.1 Many-to-many alignment

The many-to-many alignment algorithm (m2m-aligner) is based on the Many-to-Many Extension to the forward-backward algorithm proposed by Sherif [2007]. The algorithm is an extension of the forward-backward one-to-one stochastic transducer of Ristad and Yianilos [1998] which extends the original algorithm by allowing multiple graphemes to map to multiple phonemes.

The training process is based on the Expectation-Maximization (EM) algorithm presented in Algorithm 2. The training process starts with an initial mapping probability table $\delta$ which can be uniformly or randomly distributed, or else based on some distribution derived from a seed set. In the expectation step (Algorithm 2, line 4), expected counts of possible grapheme-phoneme mappings, $\gamma$, are collected from word-phoneme pairs $(s_1^I, t_1^J)$ in the training set based on the current mapping probability table $\delta$. The variables $maxS$ and $maxT$ are parameters that control the maximum sizes of grapheme-phoneme substring mappings; they can be set by using development sets. The maximization step (Algorithm 2, line 5) simply re-normalizes the expected counts to create a probability distribution. The

normalization can be perform over the whole table to create a joint distribution or per grapheme substring to create a conditional distribution. The choice of normalization can be set by using development sets. In general, for the grapheme-to-phoneme end task in the discriminative training system described in Chapter 4, the conditional distribution provides better word accuracy performance than the joint distribution. These results can be explained as the EM process with the conditional distribution creates alignments that maximize the conditional probability $P(t|s)$ of the data. These data are more suitable than the joint distribution for the discriminative systems that are finding outputs directly from the conditional probability. In the maximization step, $\delta$ probability table can be re-estimated from the expected counts, $\gamma$ as shown in Equation 3.1.

$$\delta(s,t) = \frac{\gamma(s,t)}{\sum_{t'} \gamma(s,t')} \tag{3.1}$$

The *Expectation-many2many* algorithm is presented in Algorithm 3. It first calls the two functions, *Forward-m2m* and *Backward-m2m*, to fill the $\alpha$ and $\beta$ tables. The forward probability, $\alpha$, is estimated by summing the probabilities from left to right while the backward probability $\beta$ is estimated in the opposite direction. The *Forward-m2m* algorithm is similar to lines 5 to 10 of Algorithm 3, except that it uses Equation 3.2 in line 7 and Equation 3.3 in line 10 of Algorithm 3. Similarly, the *Backward-m2m* algorithm is a symmetrical process.

$$\alpha_{i,j} \mathrel{+}= \gamma(s^i_{i-i'+1}, \_)\alpha_{i-i',j} \tag{3.2}$$

$$\alpha_{i,j} \mathrel{+}= \gamma(s^i_{i-i'+1}, t^j_{j-j'+1})\alpha_{i-i',j-j'} \tag{3.3}$$

After collecting the forward and backward probabilities, the expected counts are estimated by summing all possible grapheme-phoneme mappings in the sequence pair $(s^I_1, t^J_1)$. The expected count collected at position $i$ and $j$ is the sum of all paths that generates the sequence pair and go through $(i,j)$, divided by the sum of all paths that generate the entire sequence pair $\alpha_{I+1,J+1}$.

The EM process iteratively trains over the training set until the probabilities converge. The final many-to-many alignments are produced using the Viterbi algorithm. It finds the most likely path based on the learned probabilities as in Equation 3.4.

$$Q(0,0) = 1 \tag{3.4a}$$

$$Q(i,j) = \max_{\substack{1 \le i' \le maxS, \\ 1 \le j' \le maxT}} \begin{cases} \delta(s^i_{i-i'+1}, \_)\alpha_{i-i',j} \\ \delta(s^t_{i-i'+1}, t^j_{j-j'+1})\alpha_{i-i',j-j'} \end{cases} \tag{3.4b}$$

**Algorithm 3** Expectation many2many algorithm.

**Input:** $s_1^I, t_1^J, maxS, maxT, \gamma, \delta$
**Output:** $\gamma$
1: $\alpha \leftarrow$ *Forward-m2m* $(s_1^I, t_1^J, maxS, maxT)$
2: $\beta \leftarrow$ *Backward-m2m* $(s_1^I, t_1^J, maxS, maxT)$
3: **if** $\alpha_{I+1,J+1} = 0$ **then**
4: **return** $\gamma$
5: **for** $i = 1 \ldots I+1$ , $j = 1 \ldots J+1$ **do**
6: **for all** $i' = 1 \ldots maxS$ **such that** $i - i' \geq 0$ **do**
7: $\gamma(s_{i-i'+1}^i, \text{-}) \mathrel{+}= \frac{\alpha_{i-i',j}\delta(s_{i-i'+1}^i, \text{-})\beta_{i,j}}{\alpha_{I+1,J+1}}$
8: **for all** $i' = 1 \ldots maxS$ **such that** $i - i' \geq 0$ **do**
9: **for all** $j' = 1 \ldots maxT$ **such that** $j - j' \geq 0$ **do**
10: $\gamma(s_{i-i'+1}^i, t_{j-j'+1}^j) \mathrel{+}= \frac{\alpha_{i-i',j-j'}\delta(s_{i-i'+1}^i, t_{j-j'+1}^j)\beta_{i,j}}{\alpha_{I+1,J+1}}$
11: **return** $\gamma$

For grapheme-phoneme conversion tasks, the $maxS$ and $maxT$ are usually set to 2; therefore, the aligner constructs 1-0, 1-1, 1-2, 2-0, and 2-1 grapheme-phoneme alignments. The zero-size phoneme indicates the null phoneme and 2-2 alignments are decomposed into two 1-1 alignments. Algorithm 3, line 7 allows 1-0 and 2-0 alignments while the zero-size-of-grapheme case is excluded as an impossible mapping.

The many-to-many alignments overcome the limitation of the one-to-one assumption by relaxing the constraint of grapheme-phoneme mapping sizes. The alignments create more intuitive information, leading to better representation of training data with hidden variables. For example, the word *phoenix* [ finɪks ] is aligned as:

| *ph* | *oe* | *n* | *i* | *x* |
|---|---|---|---|---|
| f | i | n | ɪ | ks |

The substring grapheme *ph* is an example of the double grapheme problem, while the grapheme *x* is an example of the double phoneme problem (Section 2.1.1). The many-to-many alignments capture good evidence of these substring mappings and resolve such problems that exist in one-to-one alignments.

In order to incorporate the many-to-many alignments into a generation model, we require an algorithm that considers various segmentations of the grapheme input string, instead of the simple tokenization based on each grapheme as a separate unit. One possible solution is to apply a grapheme segmentation model to pre-segment words, as described in [Jiampojamarn et al., 2007]. This type of solution leads to propagation of errors due to the nature of a pipeline process. Another possible solution is to incorporate a phrase-based decoder (e.g. Zens and Ney [2004]) into the generation model, as described in [Jiampo-

jamarn et al., 2008]. The approach simultaneously searches for the most likely phonemes and grapheme segmentations avoiding the propagation of errors as in the pipeline process. These two solutions are described in depth in Chapter 4.

## 3.2 Phonetic alignment

The phonetic alignment approach was proposed in [Jiampojamarn and Kondrak, 2010]. This approach takes a different view than the EM-based approaches to the grapheme-phoneme alignment problem. Instead of aligning graphemes and phonemes as abstract symbols, the alignments are created based on the phonetic similarity between phonemes using the ALINE algorithm which was introduced to cognate identification problem [Kondrak, 2000]. ALINE was originally designed for aligning cognates, but it is sufficiently general to be used for aligning any strings of phonemes. The key idea of using this approach to grapheme-to-phoneme alignment is to represent graphemes with the phonemes that they are likely to represent. Then, the actual phonemes on the phoneme side can be aligned with the phoneme representation on the grapheme side using pure phonetic similarity.

Of course, the problem of finding the most likely phoneme for each grapheme in the first place is highly complex. However, in practice these mappings are not required to be precise. In fact, a simple method of treating every grapheme as it is used as a symbol in the International Phonetic Alphabet (IPA) [International Phonetic Association, 1999] is sufficient to approximate these mappings. The IPA is based on the Latin alphabet, but it also includes a number of other symbols. Intuitively, the 26 IPA grapheme symbols tend to correspond to the usual phonetic values that the graphemes represent in the Latin script. For languages whose orthographic systems are not based on the Latin script, a simple conversion can be performed to replace every grapheme with the IPA symbol that is phonetically closest to it [Jiampojamarn et al., 2009, 2010b].

Figure 3.1 illustrates the ALINE alignment search algorithm for the Latin word *"kentum"* and the Greek word *"hekaton"*. Each link has an associated score indicating the similarity between the two phonemes. These scores are computed based on 12 phonetic features used in ALINE. The search algorithm is based on a dynamic programming search to find the maximum of the summed link scores while maintaining monotonic links from left to right. ALINE was originally designed to create phoneme-phoneme alignments and

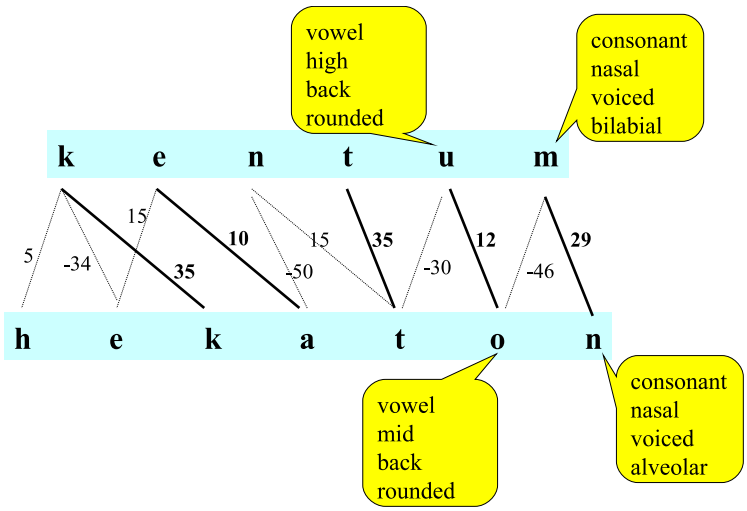


Figure 3.1: ALINE alignment example.

it does not prevent null graphemes in the source side. In order to avoid this problem in the grapheme-phoneme alignments, the following steps are taken as a post-processing algorithm. First, the algorithm attempts to remove 0-1 links by merging them with adjacent 1-0 links. If this is not possible, the algorithm then attempts to replace a pair of 0-1 and 1-1 links with a single 1-2 link where a list of allowable 1-2 links are provided as additional input. Lastly, the algorithm removes any instance that cannot be aligned. In practice, these removed instances are often annotation errors and comprise less than 1% of the entire data set.

The phonetic alignment approach thus generates 1-0, 1-1, and 1-2 links. Although the list of 1-2 links must be provided by an expert beforehand, the size of the provided list is generally small, ranging from 1 for Spanish and German to 17 for English. The solution to the double phoneme problem in this approach is more robust than the merging solution discussed in Section 2.1.1 since it only merges two phonemes when necessary.

## 3.3 Constraint-based alignment

The phonetic alignment approach incorporates phonetic features that prevent implausible links such as aligning vowels with consonants. A set of constraints is thus implicit in the phonetic similarity function. However, as mentioned, an expert is required to provide a list of possible 1-2 links in order to cover other possible links, such as the “j [ju]” and “x [ks]” cases in English. For EM-based approaches, Black et al. [1998] reported that con-

structing lists of possible phonemes for each grapheme leads to better alignment quality, resulting in an improvement in G2P conversion accuracy. The seed sets used in both the phonetic alignment and the EM-based approaches can be constructed either based purely on linguistic knowledge or in an interactive manner. These provided seed sets are in fact very important for the one-to-one alignment-based approaches because they have no ability to discover non one-to-one alignments. The many-to-many alignment approach relies on the EM algorithm to cover these links creating complex alignments.

Integer linear programming alignment and alignment by aggregation approaches were proposed in [Jiampojamarn and Kondrak, 2010] to improve the EM-based alignment approaches. I will describe the integer linear programming alignment approach in Section 3.3.1 and the alignment by aggregation approach in Section 3.3.2.

### 3.3.1 Integer linear programming alignment

The integer linear programming alignment approach applies an integer linear programming (ILP) framework to discover an optimal set of possible grapheme-phoneme mappings without a human expert. The ILP formulation aims at identifying the smallest set of grapheme-phoneme mappings that is sufficient to align all instances in the data set. The optimal set from ILP helps EM to focus on the small and sufficient set of mappings instead of automatically discovering rare mappings and assigning flat distributions [Ravi and Knight, 2009] over the grapheme-phoneme mapping probabilities. The ILP formulation employs the following constraints during its search for the optimal mapping set:

- Monotonic alignment: grapheme-phoneme mappings are created from left to right with no crossing links.
- No null grapheme: it is impossible to create a mapping of a null grapheme to a phoneme.
- Phoneme coverage: it is required that a phoneme should be linked by at least one grapheme.
- Grapheme coverage: it is required that a grapheme should be linked to at least one phoneme or a null phoneme.
- Mapping constraint: only 1-0, 1-1 and 1-2 links are allowed in the alignments.

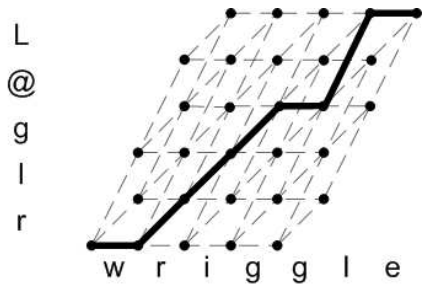


Figure 3.2: A network of possible links.

For convenience, let $G(s,t)$ and $G(s,t_1t_2)$ be the global lists of 1-1 and 1-2 grapheme-phoneme mappings being minimized and let variables $X, Y$, and $Z$ be local variables that correspond to 1-0, 1-1 and 1-2 grapheme-phoneme mappings. Both global and local variables hold binary values. The ILP formulation is stated in Equation 3.5, where 1-1 and 1-2 links are counted in the global list while there is no cost to introduce 1-0 links to allow any grapheme mapping to a null phoneme.

$$\text{minimize: } \sum_{s,t} G(s,t) + G(s,t_1t_2) \tag{3.5a}$$

$$\forall_{i,j,k} Y(i,j,k) \leq G(s_{i,k}, t_{j,k}) \tag{3.5b}$$

$$\forall_{i,j,k} Z(i,j,k) \leq G(s_{i,k}, t_{j,k}t_{j+1,k}) \tag{3.5c}$$

$$\forall_{i,j,k} \begin{array}{l} X(i,j,k) + Y(i,j,k) + Z(i,j,k) = \\ X(i-1,j-1,k) + Y(i-1,j-1,k) + Z(i-1,j-2,k) \end{array} \tag{3.5d}$$

In the lexicon entry $k, s_{i,k}$ is a grapheme at position $i$ and $t_{j,k}$ is a phoneme at position $j$. Inequalities 3.5b and 3.5c are two constraints that ensure any proposed links in local variables are counted in the global list. Any active link in each instance is included in the global list; however, active pairs in the global list do not necessarily activate local variables. Active and inactive links receive value 1 and 0 respectively so that the objective in expression 3.5a is to obtain the smallest number of active grapheme-phoneme mapping in the global variables. The constraint in Equation 3.5d forces local variable links to receive a value of 1 from left to right by ensuring the sum of the links entering each node to be equal to the sum of the links leaving each node. Figure 3.2 shows a network of possible links for the lexicon entry $k = 47$, the word "wriggle [rIg@L]". There are three 1-0 links (level), three 1-1 links (diagonal), and one 1-2 link (steep) active links in the network. Their local variables are $X(1,0,47), Y(2,1,47), Y(3,2,47), Y(4,3,47), X(5,3,47), Z(6,4,47)$, and $X(7,5,47)$ corresponding to the global variables: $G(r,r), G(i,I), G(g,g)$ and $G(l,@L)$.

In practice, allowing a full search of possible grapheme-phoneme mappings creates

complex ILP problems and gives too much freedom to the model, leading to inferior results. To reduce the number of possible grapheme-phoneme mappings, the system first trains on a subset of training data that includes only the lexicon entries with more phonemes than graphemes using the full set of variables. This model discovers a small number of possible 1-2 mappings. Then, the final ILP model limits the 1-2 links to the set found in the first pass. Finally, the model is trained on the full set of the training data to achieve the optimal set of grapheme-phoneme mappings.

The set of allowable grapheme-phoneme mappings determined by the ILP solver can also be used as an input to the EM alignment algorithm. After inducing the minimal set of grapheme-phoneme mappings, the EM model is constrained to use only those mappings with the exclusion of all others. Initially, the probabilities of the minimal mappings are set with a uniform distribution. The other mappings are forced to have zero probability. The EM training process works in a similar fashion to the many-to-many alignment algorithm presented in Section 3.1, except that the grapheme size is limited to be exactly one, and any grapheme-phoneme mapping that is not in the minimal set is forced to receive zero count during the expectation step. The Viterbi decoder performs the $\arg\max$ operation to retrieve the best alignments after the parameters converge.

### 3.3.2 Alignment by aggregation

The alignment by aggregation approach also constrains the grapheme-phoneme mapping possibilities used by its search model. The key idea is to train a one-to-many alignment model using EM which is less complex that the many-to-many alignment one to avoid rare mappings. Later, the trained model generates $n$-best alignments together with their alignment probabilities. The final alignment for each instance is created by aggregation. The aggregation process considers only the generated alignments that have a ratio between their probability values and the probability of the best alignment higher than a certain threshold.

The many-to-many aligner in Section 3.1 has the ability to create precise links involving more than one grapheme, such as *ph*:f. However, it also tends to create non-intuitive links such as *se*:z for the word *phrase* [f r e z], where $e$ is clearly a case of a "silent" grapheme. I propose an alternative EM-based alignment method that instead utilizes a list of different high-quality *one-to-many* alignments created with the algorithm described in Section 3.1 and aggregates 1-M links into M-M links in cases when there is a disagreement between alignments within the list. For example, if the list contains the two alignments shown in

Figure 3.3, the algorithm creates a single many-to-many alignment by merging the first two 1-1 links into a single *ph:f* link. However the two rightmost links are *not* merged because there is no disagreement between the two initial alignments. As a result, the aggregated alignment approach creates *s*:z and *e*:_ links which retain the generalities of alignments more than creating a single *se*:z link. That is required, if *se*:z is used, other example words to learn useful productions of *s*:z and *e*:_.

```
p h r a s e      p h r a s e
| | | | | |      | | | | | |
f _ r e z _      _ f r e z _
```

Figure 3.3: Alignment examples of *"phrase"*.

In order to generate the list of best alignments, instead of using a standard Viterbi decoding algorithm, a modified one shown in Algorithm 4 is used to generate $n$-best alignments. The algorithm maintains $n$-best scores for each stage $Q_{i,j}$ during the forward pass by appending scores to an array instead of keeping only the maximum score, as shown in lines 6 and 9. Line 7 is the relaxation part to introduce a possible mapping between grapheme $s_i$ and a substring of phonemes $t$. The $maxY$ variable is the maximum size of the substring which controls the search space of output $t$. The final $n$-best alignments can be obtained from the array $Q_{I+1,J+1}$.

The aggregation process only considers those alignments that have a ratio between their probability and the probability of the best alignment larger than a certain threshold $R$. The closer this ratio parameter value is to 1, the higher the ambiguity among the alternative alignments. An optimal ratio parameter is found by using a development set. In general, when this value is set to 0.8, the $n$-best list can be as small as the size of 10 best alignments to guarantee it includes all alignments needed in the aggregation process.

## 3.4 Evaluation

There are two possible evaluation methods for assessing grapheme-to-phoneme alignment performance. The first one is to directly evaluate alignment quality by comparing generated alignments to the gold-standard alignments annotated by experts. The second is to evaluate via the G2P performance by applying the generated alignments to a standard G2P system. For the first method, the gold-standard alignments were constructed from the core vocabulary of the Combilex data set [Richmond et al., 2009]. Combilex is a high-quality pronunciation lexicon with an explicit manual alignment by experts. The evaluated set con-

**Algorithm 4** $n$-best alignments.

**Input:** word-phoneme string: $s_1^I$, $t_1^J$ and mapping probability $\delta$
**Output:** $n$-best results of $Q_{I+1,J+1}$

```
Initialize Q = ∅
for i = 1..I + 1 do
    K = ∅
    for j = 1..J + 1 do
        for q ∈ Q_{i−1,j} do
            append q · δ(s_i, _) to K
        for j' = 1...maxY st j − j' ≥ 0 do
            for q ∈ Q_{i−1,j−j'} do
                append q · δ(s_i, t[j − j' + 1, j]) to K
        sort K
        Q_{i,j} = K[1 : N]
return Q_{I+1,J+1}
```

tains 18,145 word-phoneme pairs whose alignments contain 550 mappings, including some complex 4-1 and 2-3 types.

Each alignment approach creates alignments from unaligned word-phoneme pairs in an unsupervised fashion. The alignment quality is reported in terms of precision, recall and $F_1$ score (the harmonic mean of precision and recall). Since the gold standard contains many M-M links, any alignment method that limits the number of graphemes in a link to one cannot obtain recall higher than 90.02%. However, it is still theoretically possible to obtain 100% precision if all 1-1 links are consistent with the M-M links in the gold standard. The $F_1$ score corresponding to perfect precision and upper-bound recall is 94.75%.

It is also useful to compute the alignment entropy which was proposed by Pervouchine et al. [2009] to evaluate the quality of alignments when the gold-standard alignments are not available. The entropy indicates the uncertainty of a mapping between grapheme $s$ and phoneme $t$, using a generated corpus alignment. The formula is:

$$H = -\sum_{s,t} P(s|t) \log P(s|t) \tag{3.6}$$

The second evaluation method is to evaluate alignments via grapheme-to-phoneme conversion performance. A standard grapheme-to-phoneme conversion system is trained using the different alignments. The difference in grapheme-to-phoneme conversion performance from different models is therefore directly due to the contribution of the different alignments. In our work, we use two different grapheme-to-phoneme conversion systems as the standard systems trained with the different alignments. The first G2P system is a

| Aligner | Precision | Recall | $F_1$ score | Entropy | G2P 1-1 | G2P M-M |
|---|---|---|---|---|---|---|
| *BaseEM* | 96.54 | 82.84 | 89.17 | 0.794 | 50.00 | 65.38 |
| *ALINE* | **99.90** | **89.54** | **94.44** | 0.672 | **54.85** | 68.74 |
| *1-M-EM* | 99.04 | 89.15 | 93.84 | **0.636** | 53.91 | **69.13** |
| *IP-align* | 98.30 | 88.49 | 93.14 | 0.706 | 52.66 | 68.25 |
| *IP-EM* | 99.31 | 89.40 | 94.09 | 0.651 | 53.86 | 68.91 |
| *M-M-EM* | 96.54 | 97.13 | 96.83 | 0.655 | — | 68.52 |
| *EM-Aggr* | 96.67 | 93.39 | 95.00 | 0.635 | — | **69.35** |
| *SeedMap* | **97.88** | **97.44** | **97.66** | **0.634** | — | 68.69 |
| *Oracle* | 100.0 | 100.0 | 100.0 | 0.640 | — | 69.35 |

Table 3.1: Alignment quality, entropy, and G2P conversion accuracy on the Combilex data set.

classification-based learning system employing TiMBL [Daelemans et al., 2004], which trains the phoneme generation model using either 1-1 or 1-M alignments. The second system is the state-of-the-art online discriminative training system for grapheme-to-phoneme conversion [Jiampojamarn et al., 2008], which accepts both 1-1 and M-M types of alignments. The online discriminative training approach has shown superior results compared to the joint $n$-gram [Demberg et al., 2007], constraint satisfaction inference [Bosch and Canisius, 2006], Pronunciation-by-Analogy [Marchand and Damper, 2006], and decision tree [Black et al., 1998] approaches in the grapheme-to-phoneme conversion task on several data sets [Jiampojamarn et al., 2008]. As for training 1-1 and M-M alignments, the only difference between these alignment types is in the search component. It uses a standard Viterbi algorithm for the 1-1 case, and a phrasal decoder [Zens and Ney, 2004] for the M-M case.

The G2P performance is reported in terms of word accuracy, which rewards only completely correct phoneme outputs given test words. For Combilex, the data is randomly split into 90% for training, and 10% for testing. For all experiments, 5% of training data are held out as development data to determine when the online process should stop as well as to optimize other parameters in the system. To directly compare the results to the joint $n$-gram system of Bisani and Ney [2008], the exact data split sets were obtained for English Celex, CMUdict, NETTalk, OALD, and French Brulex. The training sizes of these data sets range from 19k to 106k words.

Table 3.1 includes the results for alignment quality (precision, recall and $F_1$ score) directly evaluated on the gold-standard alignments, as well as alignment entropy and G2P word accuracy, all on the Combilex data set. The baseline *BaseEM* is an implementation of the one-to-one alignment approach of Black et al. [1998] without a hand-built seed list.

*ALINE* is the phonetic method described in Section 3.2. *1-M-EM* is equivalent to *M-M-EM* but with the restriction that each link contains exactly one grapheme. *M-M-EM* is the m2m-aligner approach described in Section 3.1. *IP-align* is the integer linear programming alignment approach described in Section 3.3.1 without EM training. *IP-EM* is the ILP alignment approach with EM training. *SeedMap* is a hand-seeded alignment method based on a set of 377 grapheme-phoneme mappings [Jiampojamarn and Kondrak, 2010] *Oracle* is the gold-standard alignments annotated by experts in the Combilex data set.

Overall, the M-M models obtain lower precision but higher recall and $F_1$ score than the 1-1 models, which is to be expected as the gold standard is defined in terms of M-M links. *ALINE* produces the most accurate alignments among the 1-1 methods; its performance is very close to the theoretical upper bounds. Its precision is particularly impressive: on average, only one link in a thousand is not consistent with the gold standard. In terms of word accuracy, 98.97% of words have no incorrect links. Out of 18,145 words, only 112 words contain incorrect links, and a further 75 words could not be aligned. Among the 1-1 methods, *ALINE* is followed by *IP-EM*, *1-M-EM*, *IP-align*, and *BaseEM*, in that order. Among the M-M methods, EM-Aggr has slightly better precision than M-M-EM, but its recall is much worse. This is probably caused by the aggregation strategy causing EM-Aggr to "lose" a significant number of correct links. In general, the entropy measure does not mirror the quality of the alignment.

The two rightmost columns correspond to the two test G2P systems. Although better alignment quality does not always translate into better G2P accuracy, there is nevertheless a strong correlation between the two, especially for the weaker phoneme generation system (G2P 1-1). Interestingly, *EM-Aggr* matches the G2P accuracy obtained with the gold standard alignments. However, there is no reason to believe that the gold standard alignments are optimal for the G2P generation task, so that result should not be considered an upper bound but the best possible alignments obtained by human experts.

Figure 3.4 illustrates the correlation between $F_1$ score and G2P word accuracy performance in the online discriminative training system. In general, better alignment quality (i.e. $F_1$ score) leads to better G2P word accuracy. The exception is the peak in the middle of the graph. It is interesting to note that these middle pack points which include *1-M-EM*, *IP-EM*, and *ALINE* are purely 1-M models, while *EM-Aggr* is a M-M model that is constructed from the 1-M model. It is clear in the figure that the 1-1 model (*BaseEM*) achieves the lowest performance in terms of both alignment quality and G2P word accuracy. Relaxing the 1-1 constraint in the alignment model leads to statistically significant improvement.

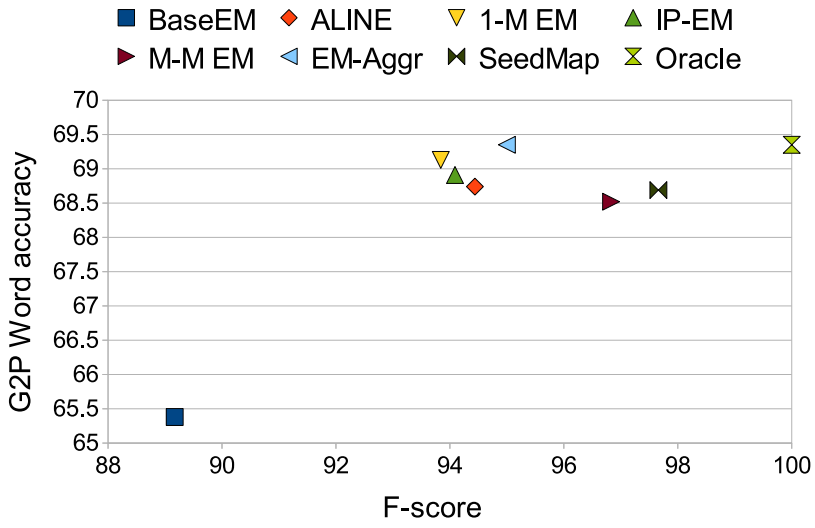


Figure 3.4: $F_1$ score versus G2P word accuracy performance.

| Aligner | Celex-En | CMUDict | NETtalk | OALD | Brulex |
|---|---|---|---|---|---|
| *BaseEM* | 75.35 | 60.03 | 54.80 | 67.23 | 81.33 |
| *ALINE* | **81.50** | 66.46 | 54.90 | **72.12** | **89.37** |
| *1-M-EM* | 80.12 | 66.66 | **55.00** | 71.11 | 88.97 |
| *IP-align* | 78.88 | 62.34 | 53.10 | 70.46 | 83.72 |
| *IP-EM* | 80.95 | **67.19** | 54.70 | 71.24 | 87.81 |

Table 3.2: G2P word accuracy using the TiMBL-based generation system.

Tables 3.2 and 3.3 show the result of G2P word accuracy performance of the standard G2P systems trained with different alignment models on English Celex, CMUDict, NETTalk, OALD and French Brulex. The TiMBL G2P generation method (Table 3.2) is applicable only to the 1-1 alignment models. *ALINE* produces the highest accuracy on four out of six datasets. The performance of *IP-EM* is comparable to *1-M-EM*, but not consistently better. *IP-align* does not seem to measure up to the other algorithms. The *BaseEM* achieves significantly lower word accuracy than the other methods, except on the NETtalk data set.

The discriminative approach (Table 3.3) is flexible enough to utilize all kinds of alignments. However, the M-M models clearly perform better than the 1-1 models. The only exception is NETTalk, which can be attributed to the fact that NETTalk already includes double-phonemes in its original formulation. In general, the *1-M-EM* method achieves the best results among the 1-1 alignment methods. Overall, *EM-Aggr* achieves the best word accuracy in comparison to other alignment methods. Except for the Brulex and CMUDict data sets, the differences between *EM-Aggr* and *M-M-EM* are statistically significant ac-

| Aligner | Celex-En | CMUDict | NETTalk | OALD | Brulex |
|---|---|---|---|---|---|
| *BaseEM* | 85.66 | 71.49 | 68.60 | 80.76 | 88.41 |
| *ALINE* | 87.96 | 75.05 | 69.52 | 81.57 | 94.56 |
| *1-M-EM* | 88.08 | 75.11 | 70.78 | 81.78 | 94.54 |
| *IP-EM* | 88.00 | 75.09 | 70.10 | 81.76 | 94.96 |
| *M-M-EM* | 88.54 | 75.41 | 70.18 | 82.43 | 95.03 |
| *EM-Aggr* | **89.11** | **75.52** | **71.10** | **83.32** | **95.07** |
| *joint $n$-gram* | 88.58 | 75.47 | 69.00 | 82.51 | 93.75 |

Table 3.3: G2P word accuracy using the online discriminative system.

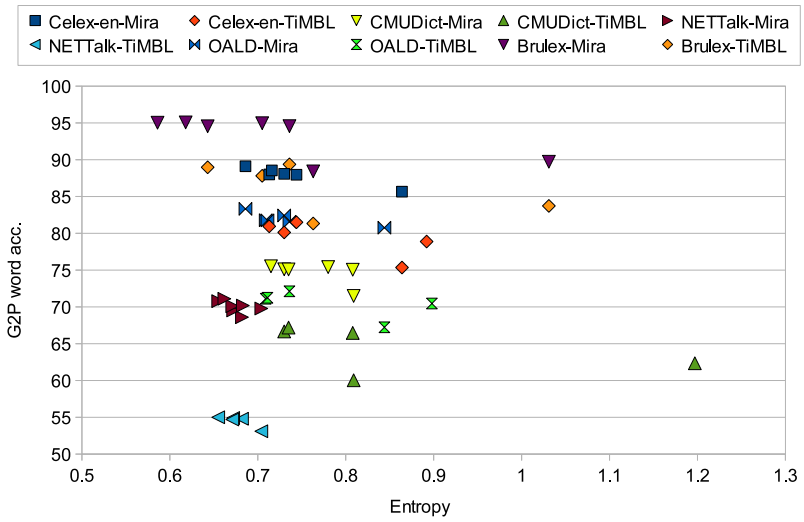


Figure 3.5: G2P word accuracy vs. alignment entropy.

cording to McNemar's test at a 90% confidence level. We also include results for the *joint $n$-gram* approach; word accuracies are taken directly from the original paper [Bisani and Ney, 2008]. *EM-Aggr* is consistently superior the joint $n$-gram approach on all sets.

Figure 3.5 contains a plot of entropy values vs. G2P word accuracy. Each point represents an application of a particular alignment method to a different data set. It appears that there is only a weak correlation between alignment entropy and G2P accuracy. So far, there is no evidence from either direct or indirect evaluations to indicate that alignment entropy is a reliable measure of grapheme-phoneme alignment quality.

## 3.5 Summary

I presented the many-to-many alignment model for the grapheme-to-phoneme conversion task. This many-to-many alignment solution advances the traditional one-to-one alignment models that have been widely used for G2P systems. The improvement of the m2m-aligner

is demonstrated in both intrinsic evaluation on gold-standard alignments and extrinsic evaluation with standard G2P systems, including classification-based and state-of-the-art online discriminative training approaches.

I also investigated several new methods for generating grapheme-to-phoneme alignments. The phonetic alignment algorithm, *ALINE*, is recommended for languages with little or no training data. The constraint-based approach achieves excellent accuracy at the cost of manual construction of seed mappings. The *ILP-EM* alignment requires no linguistic expertise and guarantees the minimal set of grapheme-phoneme mappings. The alignment by aggregation approach, *EM-Aggr*, advances the state-of-the-art results in G2P conversion. I thoroughly evaluated the resulting alignments on several data sets by feeding them into two different G2P generation systems. Finally, I employed an independently-constructed lexicon to demonstrate the close relationship between alignment quality and G2P conversion accuracy.

# Chapter 4

# Grapheme-to-phoneme generation

Chapter 3 describes methods for generating many-to-many alignments. This chapter presents the G2P generation approaches that incorporate the alignments. First, Section 4.1 proposes an approach to apply the many-to-many alignments to an existing classifier, as well as to improve the G2P performance with a language model as a post-processing step. This research is joint work with Grzegorz Kondrak and Tarek Sheif published in [Jiampojamarn et al., 2007]. A grapheme chunking model is proposed to manage double graphemes and phonemes as opposed to preprocessing with fixed lists. The language model helps the classifier to overcome its lack of awareness of previous phonemes generated in the phoneme sequence output. These approaches aim to demonstrate how many-to-many alignments can be used in widespread, existing classifiers for G2P conversion, without requiring much change in the current systems.

Later, a joint processing approach and online discriminative training framework for G2P conversion is presented in Section 4.2.1. This research was previously published in [Jiampojamarn et al., 2008]. It is a collaboration with Colin Cherry and Grzegorz Kondrak. The proposed approach unifies into one model of all components that are included in the pipeline process presented in [Jiampojamarn et al., 2007]. Phrase-based decoding is applied to replace the grapheme chunking and language models. A simple online perceptron update, based on [Collins, 2002], and a max-margin update, based on [Crammer and Singer, 2003], are applied to train the system with a large set of features, including source context $n$-gram, target transition, and linear-chain features. The experiments illustrate the state-of-the-art performance of the system, comparing to other approaches in the literature on different languages. The implementation of this system, so called "DirecTL", is publicly available as an open-source project at `http://code.google.com/p/directl-p/`.

Section 4.3 discusses the stress assignment problem in the G2P task. Most G2P systems take the stress assignment problem as a separate process and incorporate the stress assignments after the phoneme generation process [Bagshaw, 1998, Coleman, 2000]. It is an interesting research question whether stress information could help improve the overall phoneme and stress word accuracy performance. Promising results were suggested in [Black et al., 1998, van den Bosch, 1997]. This research was previously published in [Dou et al., 2009] and is joint work with Qing Dou, Shane Bergsma, and Grzegorz Kondrak. We propose several approaches for incorporating stress information to the DirecTL system, either via the output phonemes or via the input graphemes. The grapheme and phoneme stress makers are obtained using the SVM re-ranking model for stress assignment [Dou, 2009].

Section 4.4 discusses and reports results for when the DirecTL system is trained without fixed alignments from the m2m-aligner presented in Chapter 3. The system considers these alignments as hidden structures and follows the end-to-end training process of Liang et al. [2006]. This training process can be viewed as a coordinate descent algorithm that is applied in the Latent Support Vector Machine framework (LSVMs) [Felzenszwalb et al., 2008]. The algorithm alternates between finding the most likely alignments under the current feature weights, and updating the weights using the online max-margin training approach.

Finally, an integration of the joint $n$-gram features into the DirecTL system, so called "DirecTL+", is presented in Section 4.5. In order to include the joint $n$-gram features, a beam search is used instead of the exact algorithm. This revised inference procedure is adopted to accommodate the higher-order Markov features. The joint $n$-gram features create precise grapheme-phoneme evidence to the model, and they allow previous joint decisions to contribute to the current decision. This approach combines two successful methods in grapheme-to-phoneme conversion: (1) the generative joint $n$-gram model of Bisani and Ney [2008] and (2) the online discriminative training approach. The final system surpasses the performance of both the joint $n$-gram and DirecTL systems evaluated on the data sets of Bisani and Ney [2008]. This is joint work with Colin Cherry and Grzegorz Kondrak, previously published in [Jiampojamarn et al., 2010a].

## 4.1 Applying M-M alignments and a language model to G2P classifiers

One-to-one alignment approaches not only simplify the alignment training, but also create straightforward training examples for grapheme-to-phoneme generation models. Since they limit the grapheme size to be exactly one grapheme per link, the generation models can sim-

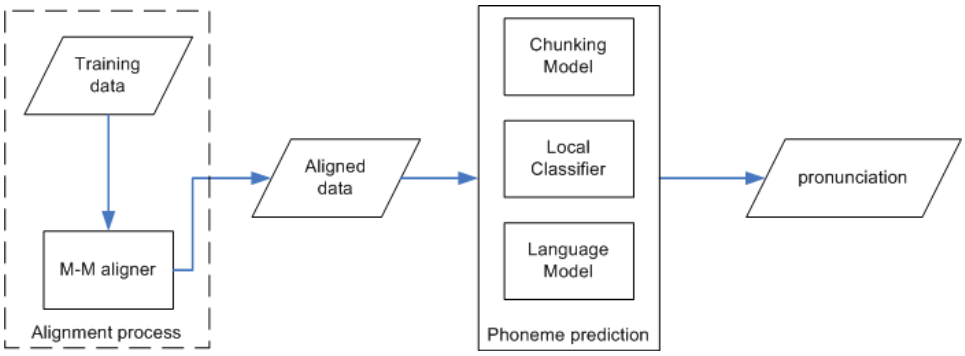


Figure 4.1: The many-to-many alignment approach framework.

ply treat each grapheme one-at-a-time during training. For these reasons, one-to-one alignments can be incorporated into an existing classifier for grapheme-to-phoneme conversion without many complications. As discussed and evaluated in Chapter 3, the many-to-many alignment approach demonstrates better alignment quality and provides more precise alignments. However, incorporating the many-to-many alignments into an existing classifier is not straightforward since the aligned grapheme is no longer a single unit. A grapheme chunking approach is described in Section 4.1.1 to solve the incompatibility between many-to-many alignments and classifiers.

In classification-based approaches, the structure of the graphemes in a word (the grapheme sequence) is well expressed in the learning models in the form of grapheme context features; however, the output pronunciation is generated without information regarding the previously generated phonemes in the sequence. Sequence-based approaches capture this phoneme sequence information well but usually treat the grapheme context as grapheme substrings tied with phoneme substrings. Therefore, the sequence-based approaches lose the ability to express the right and left context as individual graphemes.

A post-processing step to the classification-based approach described in Section 4.1.2 helps the system to capture phoneme sequence information, which is trained using phoneme sequences in the training data. The aim of this approach is to improve the existing classification-based approaches to grapheme-to-phoneme conversion, allowing them to utilize the output sequence information as well as the strong context features in the input sequence.

Figure 4.1 presents the overall framework that incorporates the many-to-many alignments described in Section 3.1 into a classifier with a language model component. Section 4.1.3 presents the successful results of this approach comparing, to standard one-to-one alignment methods and the CSInf approach of Bosch and Canisius [2006].

| $s_{i-2}$ | $s_{i-1}$ | $s_i$ | $s_{i+1}$ | $s_{i+2}$ | $chunk$ |
|---|---|---|---|---|---|
| _ | _ | so | n | g | 0 |
| _ | s | on | g | s | 0 |
| s | o | ng | s | _ | 1 |
| o | n | gs | _ | _ | 0 |

Table 4.1: An example of grapheme chunking prediction.

### 4.1.1 Grapheme chunking model

Since the many-to-many alignments are not restricted to a single grapheme token, the possible alignments for the word-phoneme sequence *phoenix* [ f i n ɪ k s ] contain grapheme substrings (chunks) aligning to phoneme substrings as:

$$\begin{array}{ccccc} ph & oe & n & i & x \\ | & | & | & | & | \\ \text{f} & \text{i} & \text{n} & \text{ɪ} & \text{ks} \end{array}$$

Although the double grapheme and double phoneme problems are intuitively solved by the alignments, a problem appears during the phoneme generation phase. How do we set grapheme boundaries in the words to generate? A simple solution of merging all graphemes that appear as grapheme chunks in the alignments is not sufficient to train a good phoneme generation model. For example, consider the words "*gash* [ g ae ʃ ]" and "*gasholder* [ g ae s h o l d ə r ]". The graphemes *sh* stay together as a grapheme chunk in the first word but are separated in the later word.

The solution to this problem is to learn a model to decide whether graphemes should be merged into grapheme chunks given only the orthographic context. By providing context information, it is possible to learn a model to identify the grapheme chunk "*sh*" and the individual graphemes *s* and *h* in the above example. The context information provides clues such as the fact the graphemes *sh* in the word *gash* stand at the end of the word, while they are in between two vowels in the word *gasholder*. Since the many-to-many alignments are set up to align graphemes and phonemes within the maximum length of 2, one can view this problem as a binary classification problem. Each grapheme in a word is evaluated to make a choice of merging with its neighbour to form a grapheme chunk or standing alone as a chunk itself. Alternatively, for alignments with a longer length, one can formulate the problem into a binary classification problem to decide if each grapheme either does or does not end a chunk [Bergsma and Wang, 2007].

Table 4.1 shows the input feature space ($s_{i-2} \ldots s_{i+2}$) and the binary output $chunk$ for the word *song* [ s ɒ ŋ z ]. The binary output value 1 indicates that the bigram grapheme $s_i$

is a chunk. In the example, the word is decomposed as *s*|*o*|*ng*|*s*, which directly corresponds to the correct phoneme sequence.

The grapheme chunking model trains on the many-to-many alignments found in the training data using an instance-based learning method [Aha et al., 1991]. If the model happens to predict consecutive overlapping chunks, only the first of the two is accepted.

### 4.1.2 Applying a language model to G2P classifiers

In classification-based approaches, contextual grapheme information is immediately available, allowing it to play a crucial role in learning the phoneme generation models. This information expresses the fact that an input word is in the form of a sequence of graphemes, which generates the phoneme output. While the structure of the input is available to the learning methods, the classifiers ignore the output structure and produce each phoneme independently. On the another hand, sequence-based models [Bisani and Ney, 2002, Chen, 2003] consider the output structure by producing each phoneme using previously predicted phonemes in the same word. However, these models generally use the grapheme context information indirectly by forming grapheme substrings tied with phoneme substrings.

The CSInf approach [Bosch and Canisius, 2006] considers the important of both directly expressing grapheme context in the classification approaches, and also of expressing the phoneme sequence in the sequence-based approaches. Unfortunately, the trigram class prediction tends to be more complex as it increases the number of target classes, while it has access to the same number of local features on the grapheme side.

An HMM approach for G2P is a poor fit to the task as shown in Taylor [2005]. The poor performance of the HMM approach is caused by its inability to condition the emission probabilities by the grapheme context. Having the state (phoneme) transition probabilities cannot recover from or compensate for the loss of the surrounding information on the orthographic side. These results suggest that phonemes depend more on graphemes than on the neighbouring phonemes (although they are also important).

Conceptually, the approach proposed here is to use a classifier to predict each phoneme given a grapheme chunk and its grapheme context. Each phoneme output from the classifier is associated with a classifier confidence value which is normalized into values between 0 and 1. In this case, the approach uses an instance-based learning technique as a local classifier to generate a set of phoneme candidates. Then, the method uses a language model to re-rank the candidate phoneme sequences. The language model serves the same purpose as the transition probabilities, while the classification confidence values serve the same

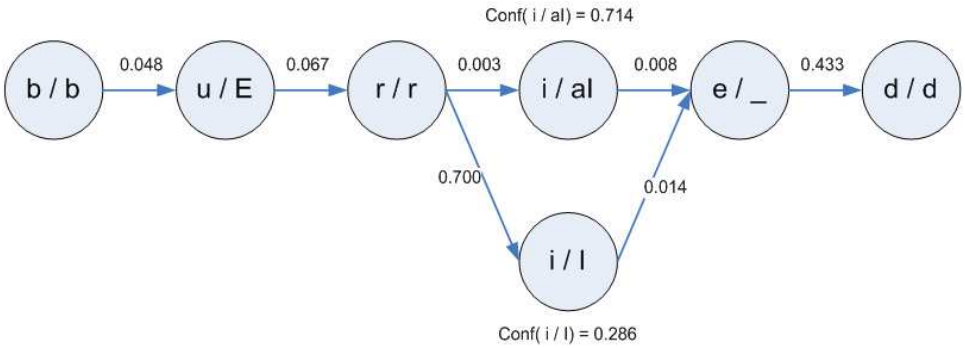


Figure 4.2: Example paths for the word *"buried"*.

purpose as the emission probabilities in HMMs. The transition probabilities are derived directly from the phoneme sequences in the training data. The optimal phoneme sequence for an input word is found with the Viterbi algorithm. It finds the most likely output sequence considering both confidence values and transition probabilities.

Figure 4.2 shows phoneme candidate paths for the word *buried*. Each node represents the generation of graphemes and phonemes. The arcs' values are first-order transition probabilities from one phoneme to the next phoneme in the path. In the example, the classifier predicts the grapheme chunk "*i*" generates phoneme [al] with a confidence value of 0.714 and generates phoneme [I] with a confidence value of 0.286. If the model is based on only the local classifier, it would produce the incorrect output of [b E r al d]. Fortunately, with the language model, the phoneme sequence [b E r I d] has a higher Viterbi score than the incorrect phoneme sequence.

### 4.1.3 Summary of evaluation and results

The proposed approaches were evaluated on CMUDict [1], French Brulex [Content et al., 1990], and German, Dutch and English CELEX corpora [Baayen et al., 1996]. Except for the English CELEX data set, all data sets are available as part of the Letter-to-Phoneme Conversion (PRONALSYL) Challenge[2]. Each data set provided by PRONALSYL is divided into 10 folds for 10-fold cross validation. For the hold-out evaluation, the first fold is designed to be the final test set, and the rest is designed to be the training set. A development set is taken from the training set. In most experiments, the second fold is designed to be the development set; thus, systems are trained on the $3^{rd}$ to $10^{th}$ folds during development. In other cases, the development set is randomly taken from 10% of the training set.

[1] WWW accessed 2008: `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`
[2] WWW accessed 2008: `http://www.pascal-network.org/Challenges/PRONALSYL/`

| Language | Data set | Number of words |
|---|---|---|
| English | CMUDict | 112,102 |
| English | NETtalk | 20,008 |
| English | CELEX | 66,189 |
| Dutch | CELEX | 116,252 |
| German | CELEX | 49,421 |
| French | Brulex | 27,473 |

Table 4.2: Number of words in each data set.

For English CELEX, The data is directly extracted from the database. After removing duplicated words, phrases, and abbreviations, the data set contains 66,189 word-phoneme pairs. The data are randomly split into 10 folds and are used to evaluate the proposed approaches in the same way as the other data sets obtained from PRONALSYL. Duplicated words are defined as instances that have the same orthographic form regardless of other information fields such as the phonemes, part-of-speech tags, stress markers, etc. Only one word-phoneme instance is included in the data set. Phrases are instances that contain a non-alphabetic character(s) on the orthographic side. Abbreviations are defined as instances that (1) contain all capital letters, (2) begin with a capital letter and consist of less than 4 letters, and (3) have a number of phonemes larger than two times the numbers of graphemes. Table 4.2 shows the number of words and language of each data set.

The local classifier for predicting phonemes is the instance-based learning implemented in the TiMBL package [Daelemans et al., 2004]. The evaluation was based on 10-fold cross validation. In the local classifier, each grapheme chunk uses context graphemes from 5 graphemes before to 5 graphemes after the focus chunk. The language model looks back 3 phonemes ($3^{rd}$ order Markov Model).

The performance comparisons are shown in Figure 4.3, reporting the average and standard deviation values of the correct word accuracy. The detailed results were published in [Jiampojamarn et al., 2007]. The 1-1 alignment model uses the same learning technique as the others but trains the model on the 1-1 alignments produced by the 1-1 epsilon scattering method [Black et al., 1998]. The HMM notation indicates the models that use the language model approach (Section 4.1.2) while the M-M alignment indicates the models that are trained on the many-to-many alignments via the bigram grapheme chunking model (Section 3.1, 4.1.1).

Clearly, impressive word accuracy improvements are achieved when the many-to-many alignments are applied as opposed to the standard approach of using 1-1 alignments. Overall, the improvements range from 2.7% to 7.6% absolute word accuracy compared to the

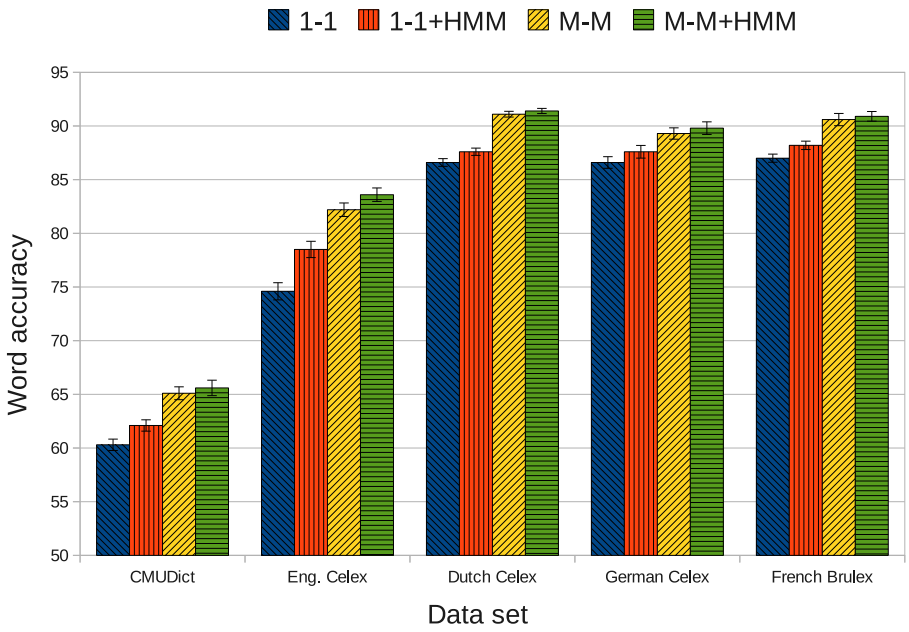


Figure 4.3: System performance showing in word accuracies on the evaluated data sets on grapheme-to-phoneme conversion task.

1-1 alignment models. Using the language model as a postprocess consistently improves performance over both 1-1 and M-M alignment models. These improvements illustrate the benefit of re-ranking the phoneme sequence candidates generated by the local classifier based on the natural sound sequences of the languages.

Table 4.3 shows results comparing to other methods on the evaluated data sets. PRONALSYS indicates the results when the system trains on the alignments provided by the challenge organizers. Their method is based on an EM one-to-one alignment approach. It is interesting to compare these results with the results of 1-1 align, which is based on the ap-

| Language | Data set | PRONALSYS | 1-1 align | CsInf | M-M+HMM |
|---|---|---|---|---|---|
| English | CMUDict | $58.3 \pm 0.49$ | $60.3 \pm 0.53$ | $62.9 \pm 0.45$ | $65.6 \pm 0.72$ |
| English | Celex | — | $74.6 \pm 0.80$ | $77.8 \pm 0.72$ | $83.6 \pm 0.63$ |
| Dutch | Celex | $84.3 \pm 0.34$ | $86.6 \pm 0.36$ | $87.5 \pm 0.32$ | $91.4 \pm 0.24$ |
| German | Celex | $86.0 \pm 0.40$ | $86.6 \pm 0.54$ | $87.6 \pm 0.47$ | $89.8 \pm 0.59$ |
| French | Brulex | $86.3 \pm 0.67$ | $87.0 \pm 0.38$ | $86.5 \pm 0.68$ | $90.9 \pm 0.45$ |

Table 4.3: Word accuracies based on 10-fold cross validation. **PRONALSYS:** Using the one-to-one alignments provided by the PRONALSYL challenge. **1-1 align:** Using the one-to-one alignment method of Black et al. [1998]. **CsInf:** Constraint satisfaction inference system [Bosch and Canisius, 2006]. **M-M+HMM:** Using the many-to-many alignment method with HMM embedded with a local prediction.

.

proach of Black et al. [1998]. Overall, 1-1 align outperforms PRONALSYS by as much as 2% in absolute word accuracies. The main difference between the PRONALSYS one-to-one alignment approach and 1-1 align is that 1-1 align does not allow null graphemes on the grapheme side. Consider the word *abomination* [ ə b ɒ m ɪ n e ʃ ə n ]: the first six graphemes and phonemes are aligned the same way by both aligners (*abomin-* [ ə b ɒ m ɪ n ]). However, the two aligners produce radically different alignments for the last five graphemes. The alignment provided by the PRONALSYS one-to-one alignments is:

| _ | _ | *a* | *t* | *i* | *o* | *n* |
|---|---|---|---|---|---|---|
| \| | \| | \| | \| | \| | \| | \| |
| e | ʃ | ə | _ | _ | _ | n |

while 1-1 align's alignment is:

| *a* | *t* | *i* | *o* | *n* |
|---|---|---|---|---|
| \| | \| | \| | \| | \| |
| e | _ | ʃ | ə | n |

Clearly, the latter alignment provides more information on how the graphemes map to the phonemes. Further improvement can be achieved by the many-to-many alignment method as it produces more precise alignments. For example, the m2m-aligner provides the correct alignment for the second part of the word *abomination*:

| *a* | *ti* | *o* | *n* |
|---|---|---|---|
| \| | \| | \| | \| |
| e | ʃ | ə | n |

Instead of adding a null phoneme in the phoneme sequence, the many-to-many aligner maps the grapheme chunk *ti* to a single phoneme.

Applying a language model as a post-processing step is based on the same hypothesis as the constraint satisfaction inference (CSInf) [Bosch and Canisius, 2006]. The results in Table 4.3 (CSInf vs. M-M+HMM) show that the HMM approach consistently improves performance over the baseline system (1-1 align), while the CSInf degrades performance on the Brulex data set. For the CSInf method, most errors are caused by trigram confusion in the prediction phase. For example, the French word "assignat [asiNa]" has possible confusion between [Na] and [na] phonemes from the trigram class prediction. The CSInf procedure produces [na] as it satisfies the most constraints on the trigram classes while the HMM model suggests [Na] because the local prediction is highly confident in grapheme "n" producing "N". In this case, both the local prediction and HMM model plays a role in the decision, while the CSInf approach over-emphasizes on the phonetic constraint satisfactions.

## 4.2 Joint processing and discriminative training

While classification-based approaches effectively provide learning models with grapheme context information, sequence-based generative approaches utilize phoneme sequence structure. For G2P, it is important to capture grapheme context when generating phoneme outputs for either single graphemes or grapheme substrings. It is also crucial to find natural phoneme sequences as output. The proposed approach to G2P is based on an online discriminative training algorithm that is capable of expressing an arbitrary number of features. The discriminative training occurs in the context of a joint processing framework that collapses the pipeline processes of grapheme chunking, phoneme generation, and sequence modeling into a single unified framework. The learning process optimizes parameters for all components simultaneously, and empirically achieves higher performance.

### 4.2.1 Joint processing

Recall that the approach described in Section 4.1 improves G2P performance by:

1. incorporating the many-to-many alignments via grapheme chunking model which allows single graphemes or grapheme substrings to generate phonemes.

2. capturing the contextual grapheme information directly in the classifier when generating phoneme.

3. using a language model to finds the optimal phoneme sequence.

The previous approach adheres to the pipeline process shown in Figure 4.4a. In general, pipeline processes are undesirable for two reasons: (1) error propagations, and (2) each process is trained without consideration of the other processes.

First, when decisions are made in sequence, errors made early in the sequence can propagate forward and throw off later processing. Second, each module is trained independently, and the training methods are not aware of the tasks performed later in the pipeline. For example, optimal parameters for a phoneme generation model may vary depending on whether or not the module will be used in conjunction with a phoneme sequence model.

A joint approach to grapheme-to-phoneme conversion is proposed by first collapsing the pipeline processes between the sequence model and the phoneme predictor illustrated in Figure 4.4b. The model is trained using an online discriminative learning method, such as an averaged perceptron HMM of Collins [2002]. The averaged perceptron HMM naturally handles the sequence modeling while it retains the capability of representing arbitrary

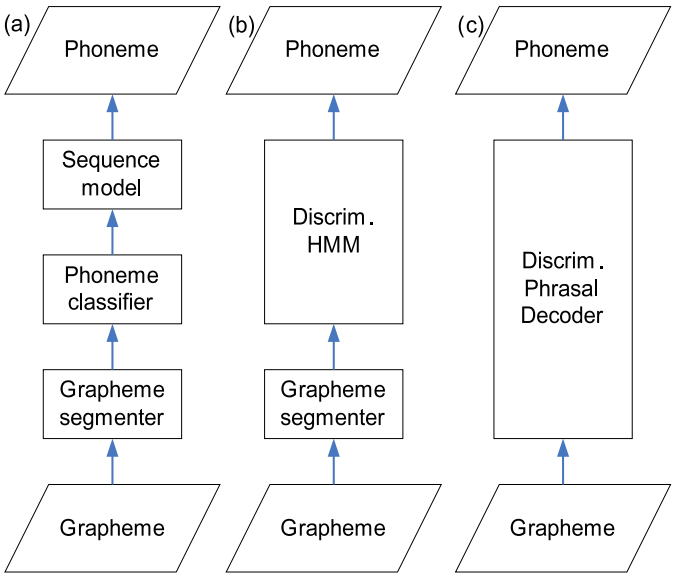


Figure 4.4: Collapsing the pipeline approach.

features that are not limited to only source-side graphemes. With the ability of expressing a rich number of features, the model can incorporate many overlapping $n$-gram features to represent grapheme context in a manner similar to that of classification-based methods. In addition, the method is free to conjoin grapheme context features with sequence phoneme features to create long dependency features as the so-called linear-chain features [Sutton and McCallum, 2006].

Next, the approach completely unifies the pipeline processes by folding the grapheme segmenter into the decoder via a monotone phrasal decoder [Zens and Ney, 2004], as shown in Figure 4.4c. The monotone phrasal decoder uses a monotone search constraint that processes graphemes and generates phonemes from left to right. This framework trains the model using the many-to-many alignments presented in Section 3.1 without committing to any specific grapheme chunk. The training process optimizes the parameters in such a way that input structures, phoneme translations, and output phoneme sequences are taken into account in a single process.

Optimizing the parameters with the perceptron update method is simple and efficient. However, one can replace the perceptron update method with the more robust Margin Infused Relaxed Algorithm (MIRA) [Crammer and Singer, 2003], which adds an explicit notion of margin to obtain better learning parameters that better separate the correct output from the incorrect ones.

**Algorithm 5** Online discriminative training.

**Input:** grapheme-phoneme example pairs $s$ and $t$.
**Output:** $\alpha$

1: $\alpha = \vec{0}$
2: **for** $K$ iterations over training set **do**
3:  **for all** grapheme-phoneme sequence pairs $(s, t)$ in the training set **do**
4:  $\hat{t} = \arg\max_{t' \in T} [\alpha \cdot \Phi(s, t')]$
5:  update weights $\alpha$ according to $\hat{t}$ and $t$
6: **return** $\alpha$

### 4.2.2 Online discriminative training

An online discriminative training process is outlined in Algorithm 5. The training process iteratively finds the best output(s) given the current weights (model parameters), and then updates the weights in such a way that the model favors the correct answer over the incorrect ones. There are three main components which will be described in detail in Section 4.2.3, 4.2.4, and 4.2.5. The first component describes how the model handles input words and output phoneme sequences, feature representations and the scoring function representing a weighted linear combination of features, $\alpha \cdot \Phi(s, t')$ in line 4. The second component describes the decoders used to perform the $\arg\max$ operation in the algorithm. The decoders are a Viterbi-based decoder for the system in Figure 4.4b and a phrase-based decoder for the system in Figure 4.4c. The last component describes the weight update algorithms; this is either the perceptron update [Collins, 2002] or the MIRA update method [Crammer and Singer, 2003].

### 4.2.3 Model

Given an input word $s$ and an output phoneme sequence $t$, let $\Phi(s, t)$ be a feature vector representing the evidence for the sequence $s$ found in $t$, and let $\alpha$ be a feature weight vector providing a weight for each component of $\Phi(s, t)$. At training time, the input word $s$ and output phoneme consist of $m$ substrings, such that $s_i$ generates $t_i$. These substrings are taken from the many-to-many alignments. At testing time, these substrings are handled by either the segmentation module or the phrase-based decoder.

Table 4.4 shows the feature template included in $\Phi(s, t)$. These features are binary features indicating whether or not the feature is present in the current $(s, t)$. The context features express grapheme evidence found in the input string $s$, centered around the generator $s_i$ of each $t_i$. The parameter $c$ is the size of the context windows. These context features include not only grapheme unigrams but all possible $n$-grams that fit within the

| | |
|---|---|
| context | $s_{i-c}$, $t_i$<br>...<br>$s_i$, $t_i$<br>...<br>$s_{i+c}$, $t_i$<br>$s_{i-c}s_{i-c+1}$, $t_i$<br>...<br>$s_{i+c-1}s_{i+c}$, $t_i$<br>...<br>$s_{i-c} \ldots s_i \ldots s_{i+c}$, $t_i$ |
| transition | $t_{i-1}$ $t_i$<br>...<br>$t_{i-M} \ldots t_{i-1}$ $t_i$ |
| linear-chain | $s_{i-c}$ $t_{i-1}$, $t_i$<br>...<br>$s_i$ $t_{i-1}$, $t_i$<br>...<br>$s_{i+c}$ $t_{i-1}$, $t_i$<br>$s_{i-c}s_{i-c+1}$ $t_{i-1}$, $t_i$<br>...<br>$s_{i+c-1}s_{i+c}$ $t_{i-1}$, $t_i$<br>...<br>$s_{i-c} \ldots s_i \ldots s_{i+c}$ $t_{i-1}$, $t_i$ |

Table 4.4: Feature template.

context windows. The transition features are HMM-like transition features that express the cohesion in the output side. The parameter $M$, called the *Markov order*, limits the number of phonemes the model can look back at. The linear-chain features [Sutton and McCallum, 2006] link phoneme transitions (e.g. $t_{i-1}$ to $t_i$) with context features.

### 4.2.4 Search

In Algorithm 5 line 4, the system must find the best possible output phoneme sequence ($\arg\max$) given the current weight vector $\alpha$. In the pipeline approach (Figure 4.4b), the input word is first segmented into grapheme substrings by the grapheme chunking model described in section 4.1.1. The search for the best possible phoneme sequence can then be performed effectively by the standard HMM Viterbi algorithm. In the joint approach 4.4c, the grapheme segmentation and phoneme generation are simultaneously performed using a monotonic phrasal decoder [Zens and Ney, 2004]. The search enumerates possible grapheme segmentations and their output phonemes together. This can be perform efficiently via dynamic programming. The dynamic programming recurrence equations are

shown in Equation 4.1. The table $Q(g, p)$ keeps the maximum score of the phoneme sequence ending with the phoneme $p$, generated by the grapheme sequence $s_1 \dots s_i$. Since the grapheme chunk information is not provided to the model in this case, this framework views $s$ as a sequence of $G$ graphemes instead of substrings. The phoneme $t'$ is the phoneme produced in the previous step. The expression $\phi(s^i_{i'+1}, t', t)$ is a convenient way to express the subvector of our complete feature vector $\Phi(s, t)$ that describes the substring pair $(s_i, t^i_{i-1})$, where $s_i = s^g_{g'+1}$, $t_{i-1} = p'$ and $t_i = p$. The value $N$ limits the size of the dynamically created substrings. The special symbol \$ represents a starting phoneme or ending phoneme. The value in $Q(G + 1, \$)$ is the score of highest scoring phoneme sequence corresponding to the input word. The actual sequence can be retrieved by backtracking through the table $Q$.

$$\begin{aligned} Q(0, \$) &= 0 \\ Q(g, p) &= \max_{\substack{p', p, \\ g-N \le g' < g}} \{\alpha \cdot \phi(s^g_{g'+1}, t', t) + Q(g', p')\} \\ Q(G + 1, \$) &= \max_{p'} \{\alpha \cdot \phi(\$, p', \$) + Q(G, p') \end{aligned} \tag{4.1}$$

### 4.2.5 Online updates

The update step in line 5 of the training algorithm can be performed by the perceptron update [Collins, 2002]. The updates are relatively simple, invoking only adding and subtracting vector operations. The weight vector $\alpha$ is updated according to the best output $\hat{t}$ found under the current $\alpha$ and the true answer $t$ found in the training data. If the output $\hat{t}$ is correct, there is no update to the weight vector $\alpha$. Otherwise, the weight vector is updated by subtracting the feature vector of the input $s$ and the wrong answer $\hat{t}, \Phi(s, \hat{t})$, and adding the feature vector of the input $s$ and the true answer $t, \Phi(s, t)$, as shown in Equation 4.2.

$$\alpha = \alpha + \Phi(s, t) - \Phi(s, \hat{t}) \tag{4.2}$$

The training process is iteratively for $K$ iterations, which is determined as the point where a decline performance occurs on the held-out set. Conceptually, if the problem is separable, the perceptron is guaranteed to find an $\alpha$ such that:

$$\forall \hat{t} \in T - \{t\} : \alpha \cdot \Phi(s, t) > \alpha \cdot \Phi(s, \hat{t}) \tag{4.3}$$

However, most problems are not separable. The average of all $\alpha$ values throughout the training process is used in place of the $\alpha$ from the final training iteration in order to obtain better generalization to unseen data [Collins, 2002].

In the perceptron update, the model is trained by seeing errors made by the current weight vector $\alpha$. There is no update to the model so long as the system predicts the correct phoneme sequence under the current model. Each update takes penalties of the wrong answer $(\Phi(s, \hat{t}))$ and rewards the correct answer of $(\Phi(s, t))$ from the current weight vector. In other words, the perceptron has no notion of margin, measure of how well the vector $\alpha$ separates the true answer from incorrect ones.

To address this, the Margin Infused Relaxed Algorithm (MIRA) [Crammer and Singer, 2003] updates the model based on the system's $n$-best outputs. It employs a margin update so that the new weight vector $\alpha$ separates the true answer from those incorrect answers by at least as much as the structured loss. The loss function provides the cost of producing a given wrong output. The update process can be described as an optimization problem:

$$\begin{aligned} &\min_{\alpha_n} \parallel \alpha_n - \alpha_o \parallel \\ &\text{subject to } \forall \hat{t} \in T_n : \\ &\alpha_n \cdot (\Phi(s, t) - \Phi(s, \hat{t})) \geq \ell(t, \hat{t}) \end{aligned} \tag{4.4}$$

where $T_n$ is a set of $n$-best outputs found under the current model, $t$ is the correct answer, $\alpha_o$ is the current weight vector, $\alpha_n$ is the new weight vector, and $\ell(t, \hat{t})$ is the loss function. The loss function here can be described as the 0-1 loss function where $\ell(t, \hat{t}) = 0$ if $t = \hat{t}$, otherwise $\ell(t, \hat{t}) = 1$. The 0-1 loss function is appropriate because it has the same objective function as the word accuracy evaluation metric. The output phoneme is judged to be wrong when the output phoneme sequence and the true answer are not identical. Other possible loss functions could be based on the Levenshtein distance between $\hat{t}$ and $t$, called the *"phoneme loss"*, and the combination of the 0-1 loss and the phoneme loss functions.

MIRA training is similar to averaged perceptron training, but instead of finding the single best answer, it finds the $n$-best answers ($T_n$) and updates weights according to Equation 4.4. To find the $n$-best answers, the Viterbi decoding and monotone search algorithms are modified to keep track of the $n$-best phonemes at each cell in the dynamic programming. The optimization in Equation 4.4 is a standard quadratic programming problem that can be solved by using Hildreth's algorithm [Censor and Zenios, 1997].

### 4.2.6 MIRA implementation

For convenience, the MIRA update algorithm shown in Equation 4.4 can be implemented using the SVM$^{light}$ framework [Joachims, 1999]. SVM$^{light}$ provides the quadratic program solver shown in Equation 4.5.

$$\begin{array}{l} \min_{w,\xi} \frac{1}{2} \parallel w \parallel^2 + C \sum_i \xi_i \\ \text{subject to } \forall i, \\ w \cdot z_i \geq rhs_i - \xi_i \end{array} \tag{4.5}$$

In order to approximate a hard margin using the soft-margin optimizer of SVM$^{light}$, a very large penalty value is assigned to the parameter $C$, thus making the use of any slack variables ($\xi_i$) prohibitively expensive. The vector $w$ is defined as the difference between the new and previous weights: $w = \alpha_n - \alpha_o$, so that the minimization objective in both equations, Eq. 4.5 and Eq. 4.4, are the same. Since each $\hat{t}$ in the $n$-best list $(T_n)$ needs a constraint based on its feature difference vector, $z_i$ is defined as:

$$\forall \hat{t} \in T_n : z_i = \Phi(s,t) - \Phi(s,\hat{t})$$

Substituting that equation along with the inferred equation $a_n = a_o + w$ into the original MIRA constraints yields:

$$(\alpha_o + w) \cdot z_i \geq \ell(t,\hat{t})$$

Moving $\alpha_o$ to the right-hand-side of the equation to isolate $w \cdot z_i$ on the left, the constraints in both equations are now the same.

In summary, the MIRA update algorithm is implemented in SVM$^{light}$'s optimizer by setting:

| SVM$^{light}$ | MIRA |
|---|---|
| $w$ | $\alpha_n - \alpha_o$ |
| $z_i$ | $\Phi(s,t) - \Phi(s,\hat{t})$ |
| $rhs_i$ | $\ell(t,\hat{t}) - \alpha_o \cdot z_i$ |

The output of the SVM$^{light}$ optimizer is an update vector $w$ to be added to the current $\alpha_o$. The SVM$^{light}$ optimizer is called per each training example in the update step in Algorithm 5, line 5.

### 4.2.7 Summary of evaluation and results

Experiments were conducted to evaluate the proposed approaches using 9-fold cross validation on the training set of English CELEX [Baayen et al., 1996]. Note that 10% of the corpus is reserved to be the final test set for comparison. More details of evaluation and

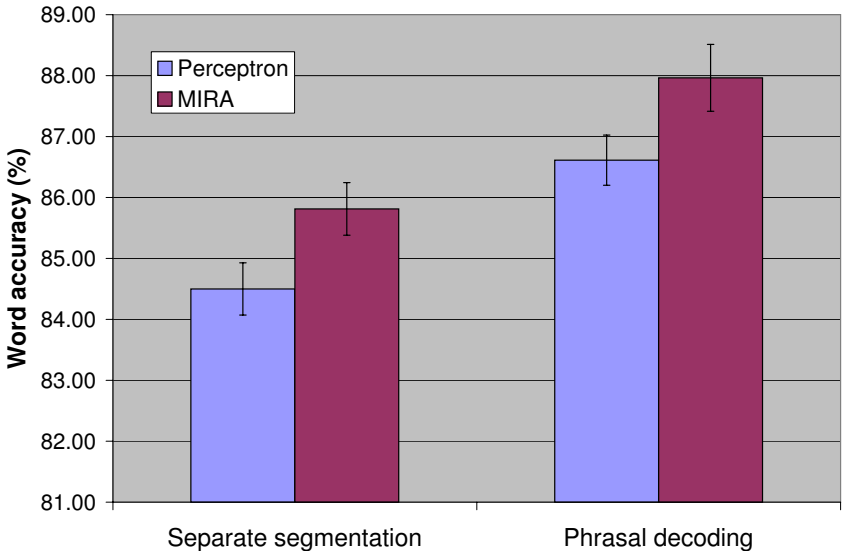


Figure 4.5: Separate segmentation versus phrasal decoding in terms of average word accuracy and standard deviation.

results can be found in [Jiampojamarn et al., 2008]. From the preliminary experiments on the development set, increasing the context size has a dramatic effect on accuracy, but the effect begins to level off for context sizes greater than 5. Henceforth, for all experiments reported here, the context features have a context window size equal to 5. Similarly, for MIRA updates, a large improvement of 2% absolute word accuracy is achieved by increasing the size of the $n$ best list from 1 to 5. However, varying the size of the $n$-best list from 5 to 50 has an insignificant effect on accuracy. Therefore, the MIRA models use an $n$-best list of length 10 to update the feature weights, based on the loss function that combines 0-1 and phoneme error rate, due to this function's marginal improvement over each individual loss function.

Figure 4.5 shows the system performance in terms of word accuracy after adding the option to conduct joint segmentation through phrasal decoding. The 15% relative error deduction demonstrates the utility of folding the segmentation step into the search. It also shows that the joint framework enables the system to reduce and compensate for errors that occur in a pipeline. By replacing the perceptron update with the MIRA update, the system obtains the improvement of 7% relative improvement in word accuracy. This improvement illustrates the more powerful update strategy of MIRA. MIRA updates the feature weights based on an $n$-best list rather than simple subtraction and addition operations as in the perceptron.

Figure 4.6 shows the effect of the sequence features on the MIRA system. Adding the

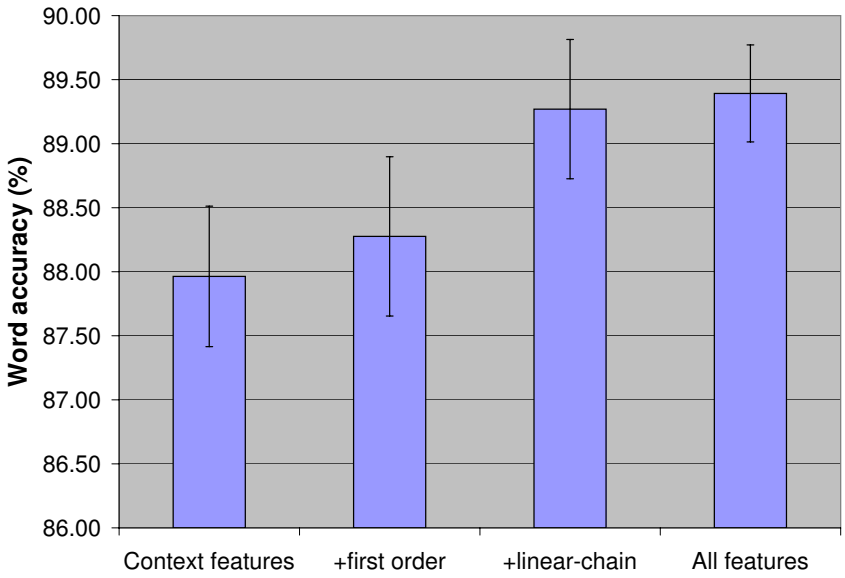


Figure 4.6: The effect of sequence features on the joint system in terms of average word accuracy and standard deviation.

first order HMM features contribute little improvement by itself, but combined with the more powerful linear-chain features achieve a relative error reduction of 12% in word accuracy. In general, the linear-chain features make a much larger difference than the relatively simple transition features, which underscores the importance of using source-side context when assessing sequences of phonemes.

Figure 4.7 shows the system's learning curve when evaluated on the development set with different amounts of training data. The sizes of the training data range from 5K to 57K words (10% - 100% of the training data). Overall, the system performance improves linearly, when the training size is increased exponentially. At the 50% training data point, the system achieves 83.19% word accuracy for its top-1 output, while the oracle selector from the 2-best answers achieves 90.85% word accuracy. This oracle word accuracy is higher than the top-1 word accuracy of the system that trains on the full training data set (89.72% vs. 90.85%). It suggests that the system requires approximately twice the training data to optimally the 2-best outputs. Although the graph does not show the convergence point for the system performance with respect to the training data size, it is interesting to observe that the system obtains 98% word accuracy with the oracle 50-best answers in systems that train with both 50% and with all of the training data.

Table 4.5 shows a comparison of different approaches on the evaluation sets including Dutch CELEX (D.CELEX), German CELEX (G.CELEX), English CELEX (E.CELEX),

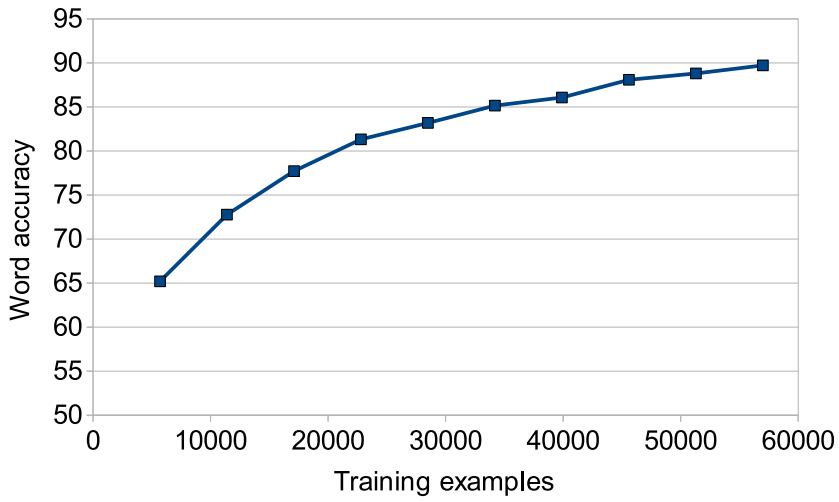


Figure 4.7: Word accuracy on the development set showing the learning curve of the system trained with different amounts of training data.

| Corpus | DirecTL | M-M+HMM | Joint n-gram* | PbA* | CART* | SMT* |
|---|---|---|---|---|---|---|
| E.CELEX | **90.51%** | 84.81% | 76.3% | - | - | - |
| D.CELEX | **95.32%** | 91.69% | - | - | - | 91.63% |
| G.CELEX | **93.61%** | 90.31% | 92.5% | - | 89.38% | 90.20% |
| Nettalk | **67.82%** | 59.32% | 64.6% | 65.35% | - | - |
| CMUDict | **71.99%** | 65.38% | - | - | 57.80% | 63.81% |
| Brulex | **94.51%** | 89.77% | 89.1% | - | - | 86.71% |

Table 4.5: Comparison of word accuracy on the evaluation sets. **DirecTL**: Online discriminative training framework. **M-M+HMM**: Many-to-Many HMM system (Section 4.1) **Joint n-gram**: Joint n-gram model [Demberg et al., 2007]. **PbA**: Pronunciation by Analogy [Marchand and Damper, 2006]. **CART**: CART decision tree system [Black et al., 1998]. **SMT**: Phrase Based Statistic Machine Translation approach for G2P [Rama et al., 2009]. The columns marked with * contain results reported in the literature. "-" indicates no reported results.

Nettalk, CMUDict, and French Brulex data sets. DirecTL refers to the full system with the discriminative training approach and MIRA updates, and the phrasal decoder with context, transition and linear-chain features. All parameters, including the size of the $n$-best list, the amount of grapheme context, and the choice of loss functions, were established on the English CELEX development set, as presented in the previous experiments. Except for the M-M+HMM results, all other results are taken directly from their original publications. M-M+HMM refers to the system described in Section 4.1 and is evaluated directly on the data sets. The joint $n$-gram, Pronunciation by Analogy and CART models are described and discussed in Chapter 2. SMT refers to the results of Rama et al. [2009] who apply a standard phrase-based statistical machine translation system to the task. They use GIZA++ [Och and

Ney, 2003] for aligning graphemes with phonemes in the training sets and train the models using the minimum error rate training algorithm [Och, 2003] with the A* beam search decoder [Koehn et al., 2003]. The implementation of the SMT model above is available as part of the MOSES toolkit [Koehn et al., 2007].

Although these comparisons are necessarily indirect due to different experimental settings, they strongly suggest that DirecTL outperforms all other results on all data sets, in some case by large margins. The joint $n$-gram approach previously reported the best result for German CELEX of 92.5% and PbA showed the best result for English Nettalk of 65.35%. The best results on the other sets were previously achieved by the M-M+HMM approach. DirecTL achieves from 3% to 8% absolute word accuracy improvement over the M-M+HMM approach and advances the previously best results of German CELEX and English Nettalk by as much as 2% in absolute word accuracy improvement. These experimental results demonstrate the power of a joint approach and online discriminative training with a large set of features.

## 4.3 Stress markers combination

In many languages, for example, English, German and Dutch, word pronunciations include "stress".Stress helps humans recognize and differentiate spoken words, as stress empathizes certain syllables in duration, pitch and loudness. Stress markers are usually associated with word syllables. To make stress assignment an isolated task apart from syllabification, stress makers can be placed on vowel phonemes without knowing the true syllable boundaries [Webster, 2004]. Later, if the syllable boundaries are known, the mapping from the stressed vowel to the corresponding syllable is straightforward. Although stress is a prosodic feature and placed on the phoneme side, it may be useful to place stress markers on the orthographic form as well, the corresponding graphemes that produce the stress vowel in the words. This information could help further improve G2P conversion accuracy.

A SVM ranking approach for stress assignment of Dou et al. [2009] is capable of placing stress markers on both orthographic and phonemic forms without requiring true syllable boundaries. The approach first approximates syllables by splitting a word into a sequence of substring units. The substring unit includes exactly one vowel and at least one neighbouring consonant – e.g. "overdo $\rightarrow$ ov-ver-do". Since, in the phonetic form, the number of vowels is equal to the number of syllables, this simple splitting method guarantees that the correct number of syllables is generated, regardless of the true syllable boundaries. In the ortho-

graphic form, this method may result in a different number of syllables – e.g. "pronounce $\rightarrow$ ron-no-un-ce"; however, it is sufficient for the learning model. The substring units are encoded with 0 for unstressed, 1 for primary stress and 2 for secondary stress. These encoded strings are called "stress patterns". A prediction of the stress patterns is trained using a SVM model [Joachims, 2002], which assigns stress markers to substring units (vowels) given new words.

Various methods have previously been used for combining stress and phoneme generation. The simplest and most straightforward is to train a phoneme generation model without regard to stress. Then, stress markers are assigned as a post-process [Bagshaw, 1998, Coleman, 2000]. However, both van den Bosch [1997] and Black et al. [1998] argued that stress should be predicted at the same time as phonemes. They expand the output set to distinguish between stressed and unstressed phonemes. Similarly, Demberg et al. [2007] produce phonemes, stress, and syllable-boundaries within a single joint n-gram model and enforce a phonological constraint — one primary stress per word. Pearson et al. [2000] generate phonemes and stress together by jointly optimizing a decision-tree phoneme-generator and a stress predictor based on stress pattern counts. The counts help the model to generate the most common stress patterns avoiding unlikely stress patterns in the phoneme output. In contrast, Webster [2004] first assigns stress to graphemes, creating an expanded input set, and then predicts both phonemes and stress jointly. The system marks stress on grapheme vowels by determining the correspondence between affixes and stress in written words.

Based on the above inspirations, DirecTL can incorporate stress makers in the following ways:

- *Joint*: the system's inputs are graphemes and the output sequences are phonemes with stress information. This can be accomplished by creating two sets of vowels that are (1) stressed vowels and (2) unstressed vowels. This approach is a baseline system which does not incorporate an additional stress model nor does it incorporate linguistic to the stress prediction model. It simply extends the output space and learns the phoneme and stress generation based on the training data.

- *Joint+Constr*: the same as *Joint*, except that the system only outputs the stress patterns that were observed in the training data. This constraint ensures that the final outputs have valid stress patterns and avoids invalid output sequences, such as containing more than one primary stress or introducing a secondary stress without a primary one. Ultimately, the training process of this system is the same as *Joint* except

| System | Eng | | Ger | Dut |
|---|---|---|---|---|
| | $P+S$ | $S$ | $P$ | $P$ |
| *Joint* | 78.9 | 80.0 | 86.0 | 81.1 |
| *Joint+Constr* | 84.6 | 86.0 | 90.8 | 88.7 |
| *PostProcess* | 86.2 | 87.6 | 90.9 | 88.8 |
| *LexicalStress* | 86.5 | 87.2 | 90.1 | 86.6 |
| *OracleStress* | 91.4 | 91.4 | 92.6 | 94.5 |
| Festival | 61.2 | 62.5 | 71.8 | 65.1 |

Table 4.6: Combined phoneme and stress prediction word accuracy (%) for English, German, and Dutch. $P$: predicting primary stress only. $P+S$: primary and secondary.

when it generates $n$-best outputs, and when it now filters any sequence containing an invalid stress pattern.

- *PostProcess*: the system is trained to generate only phonemes. It places stress makers on the output phonemes as a post processing step using the SVM ranking model [Dou, 2009]. This is a pipeline process which uses the phoneme generation of DirecTL and the stress assignment of the SVM ranker.

- *LexicalStress*: the system's inputs are graphemes with added stress on their orthography. The orthographic stress is obtained by the SVM ranking model [Dou, 2009]. The output is phonemes with stress information. This approach shows the contribution of automatic lexical stress assignment to the full grapheme-phoneme conversion task.

- *OracleStress*: the same as *LexicalStress*, except the system has the true stress marked on the graphemes. The oracle stress markers are obtained by mapping the stressed vowels back to their corresponding graphemes via alignments in the training.

Table 4.6 shows the results of integrating stress markers into the DirecTL system. Festival's results are obtained by using the popular Festival Speech Synthesis System[3]. Overall, the *Joint* approach, which simply expands the output set, is 4%-8% worse than all other comparison systems across the three languages. These results clearly indicate the drawbacks of predicting stress using only local information. In English, both *LexicalStress* and *PostProcess* perform best, while *PostProcess* and *Joint+Constr* systems are highest on German and Dutch. Although it is inconclusive which approach is the best way to incorporate stress information into the G2P system, the *OracleStress*'s results suggest promising improvements when accurate lexical stress is provided to the generation system. Results using

[3] `http://www.cstr.ed.ac.uk/projects/festival` version 1.96, 2004

oracle lexical stress show that given perfect stress assignment on graphemes, phonemes and stress can be predicted very accurately, in all cases above 91%. The accuracy of Festival is much lower even than the baseline *Joint* approach. These results show that the DirecTL system can outperform a widely-used speech synthesis system in the generation task, as DirecTL exploits a much more powerful discriminative training model.

## 4.4 Training without alignments

Grapheme-phoneme alignments play a crucial role in the G2P training processes. The alignments between graphemes and phonemes in the training data allow phoneme generation models to train on fixed structures. By constructing the fixed structures in the training data, the phoneme generation model has only one objective to learn: to produce phoneme sequences corresponding to the given input structures. In Section 4.2, the joint processing framework illustrates the benefit of folding the grapheme segmentation process into the phoneme generation model by using phrasal decoding. Thus, the joint processing framework has two objectives: (1) to find the most likely phoneme sequences, and (2) to find the most likely input structures. It learns these structures via the alignments in the training data. The key issue here is that the alignments are automatically generated by the m2m-aligner. Although these alignments are shown to be more accurate for training G2P systems than the one-to-one alignments, the many-to-many alignments are not perfect. The learning model accepts these imperfect alignments as gold-standard structures for training.

Liang et al. [2006] proposed three different update strategies to find both hidden and output structures for machine translation. In the context of machine translation, the outputs are translations ($t$) in a target language corresponding to input sentences ($s$) in a source language. The hidden structures ($h$) are alignments between words in the source and target languages. The first strategy, called *"bold updating"*, updates feature weights towards the correct output $t$ and the highest-scoring option of all hidden structures $h$, where $h$ is generated based on the current weight parameters, input, $s$, and correct output, $t$. The update step is skipped if the current model cannot find any possible hidden structure that is a production of the input $s$ and output $t$. The second strategy, called *"local updating"*, updates the feature weights towards the highest-scoring answer in the $n$-best list. The third strategy is a combination of the first two options. It uses the first strategy by default, and the second strategy when the model cannot find the hidden structure. In machine translation, many hidden structures are unreachable due mostly to the limits of the distortion allowed in

the alignment model. The bold updating is more problematic than other strategies when a limited-distortion decoder is used in their system. With the flexibility of translation phrase swaps, they found that the local updating performs best among the three approaches in their phrase-based machine translation system.

The $\arg\max$ and update formulations in Algorithm 5 can be re-written to accommodate hidden structure as in Equations 4.6 and 4.8. Here, $h'$ is the most probable alignment between the input grapheme string $s$ and the most probable output phoneme string $\hat{t}$.

$$(\hat{t}, \hat{h}) = \arg\max_{t',h'}[\alpha \cdot \Phi(s, t', h')] \tag{4.6}$$

$$h^* = \arg\max_{h}[\alpha \cdot \Phi(s, t, h)] \tag{4.7}$$

$$\begin{array}{l} \min_{\alpha_n} \parallel \alpha_n - \alpha_o \parallel \\ \text{subject to } \forall \hat{t} \in T_n : \\ \alpha_n \cdot (\Phi(s, t, h^*) - \Phi(s, \hat{t}, \hat{h})) \geq \ell(t, \hat{t}) \end{array} \tag{4.8}$$

The alignments $\hat{h}$ are implicitly found by the phrasal decoder during its search for the maximum score of the dot product between the feature weights $\alpha$ and the feature indicators $\Phi$. These alignments are in fact the generation outputs, $\hat{t}$, of each substring unit $s_1 \ldots s_I$ that yields the maximum model score. When training with alignments using the m2m-aligner, $h^*$ are the alignments discovered by the aligner. The are fixed throughout the online training process. When training without pre-defined alignments, $h^*$ are the best alignments found at each update based on the current feature weights $\alpha$ given $s$ and the correct $t$ in the training data. The alignments for an example $s$ and $t$ can be found by $\hat{h} = \arg\max_{h' \in H}[\alpha \cdot \Phi(s, t, h')]$. Essentially, the model searches for the best substring segmentation of grapheme string $s \rightarrow s_1 \ldots s_I$ and phoneme string $t \rightarrow t_1 \ldots t_I$ . This process can be performed by a similar dynamic program to the one used by the phrase-based decoder presented in Section 4.2.4, except that the input string $s$ and the output string $t$ are fixed in the search. Before each update step, new alignments $h^*$ for an example $s$ and $t$ are found based on the current weight parameters $\alpha_o$ and they are fixed during the process of updating weights $\alpha_n$. This alternate training process is similar to a coordinate descent algorithm applied in Latent Support Vector Machines (LSVMs) [Felzenszwalb et al., 2008].

Table 4.7 shows the results of bold and local updates for the DirecTL systems. The possible alignments are initialized with a list of grapheme-phoneme mappings from the m2m-aligner. DirecTL in the table refers to the model that trains on fixed structures as presented in Section 4.2. Unlike in machine translation, the bold updating achieves better performance than the local strategy due to the fact that grapheme-phoneme alignments

| Method | Word accuracy |
|---|---|
| DirecTL | 87.85% |
| Bold update | 86.54% |
| Local update | 79.42% |

Table 4.7: G2P word accuracy of DirecTL, bold and local updates on the English CELEX development set.

| Method | Word accuracy |
|---|---|
| Full list of grapheme-phoneme mappings | 86.54% |
| without the list of mappings | 86.13% |
| without 1-m mappings | 86.28% |
| without m-1 mappings | 86.49% |

Table 4.8: G2P word accuracy of bold update approach given different alignment initialized mappings.

in G2P are monotonic and are not affected by alignment distortion. The bold update experiments suggest that given the example pairs the model always finds alignments under the current feature weights; therefore, there is no skipping case for G2P. In general, training without fixed alignments achieves lower performance than the DirecTL system with fixed alignments found by m2m-aligner. One might observe that our method of training without alignments is actually not as general as Liang et al. [2006]'s approach because the possible grapheme-phoneme pairs are given in advance (restricted to the pairs returned by m2m-aligner). Additional experiments were set up to demonstrate the effect of the pair-list initialization. When the system does not use the list of grapheme-phoneme mappings from the m2m-aligner, the word accuracy performance slightly decreases to 86.13%. The differences between using a full list of grapheme-phoneme mappings and lists of partial mappings are negligible (Table 4.8).

## 4.5 Integrating joint $n$-gram features into DirecTL

The online discriminative training framework presented in Section 4.2 improves grapheme-to-phoneme conversion accuracies by using a phrase-based decoder, and an online max-margin training regime over a large set of features. The $n$-gram context and linear-chain features capture rich information over the source and target strings. With these features, the model can use a first order Markov assumption without any significant decrease in performance. In other words, increasing the Markov order does not significantly improve the performance. This discovery at first seems to contradict results of the joint $n$-gram model of Bisani and Ney [2008], who showed that the learning model for grapheme-to-phoneme

| | |
|---|---|
| joint $n$-gram | $s_{i+1-n}t_{i+1-n}s_it_i$<br>$\dots$<br>$s_{i-1}t_{i-1}s_it_i$<br>$s_{i+1-n}t_{i+1-n}s_{i+2-n}t_{i+2-n}s_it_i$<br>$\dots$<br>$s_{i-2}t_{i-2}s_{i-1}t_{i-1}s_it_i$<br>$\dots\dots$<br>$s_{i+1-n}t_{i+1-n}\dots s_{i-1}t_{i-1}s_it_i$ |

Table 4.9: Joint $n$-gram feature template.

conversion requires at least a **fifth** order Markov assumption. In fact, their best results were with an **eighth** order Markov model. The joint $n$-gram approach takes information over source and target substrings simultaneously, increasing the Markov order not only means adding dependencies over the phonemic (target) substrings but also over the orthographic (source) substrings. Although the joint $n$-gram models are capable of capturing context information on both the source and target side, they cannot selectively use only source or target information, nor can they consider arbitrary sequences within their context window, as they are limited by their back-off schedule.

In the DirecTL model, the set of indicator features (Table 4.4) include (1) context features, (2) transition features, and (3) linear-chain features. The context features allow the model to selectively use information from the source side. The transition features describe the cohesion of the target string, while the linear-chain features allow the model to look at both source and target information jointly; however, these features do not go beyond two target phonemic substrings, $t_{i-1}, t_i$. By integrating joint $n$-gram features into the online discriminative training framework, the system not only enjoys rich context features and long-range linear-chain features, but also can now take advantage of wide joint information between the source and target substring pairs. The joint $n$-gram feature template is shown in Table 4.9.

An alternative method to incorporate a joint $n$-gram feature would be to compute the generative joint $n$-gram scores, and supply them as real-valued features to the model. As all of the other features in the DirecTL framework are indicators, the training algorithm may have trouble scaling an informative real-valued feature. Therefore, a binary feature representation is used for these joint $n$-gram features, indicating whether the model has seen particular strings of joint evidence in the previous $n-1$ operations. In this case, the system learns a distinct weight for each substring of the joint $n$-gram instead of a weight for each generative joint $n$-gram probability.

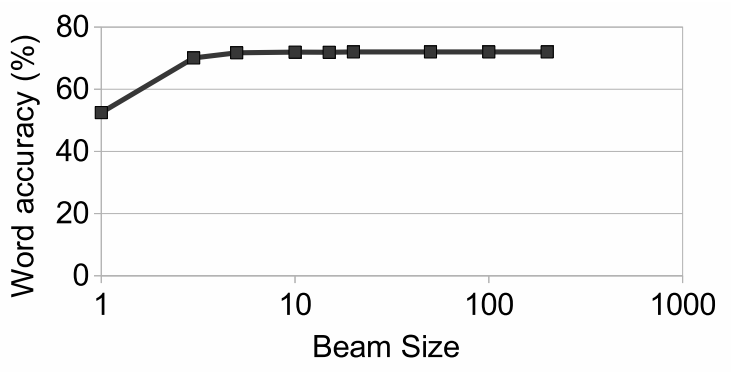


Figure 4.8: System accuracy as a function of the beam size.

In order to accommodate higher-order joint $n$-grams, a beam search is used instead of an exact phrase-based decoder presented in Section 4.2.4.A beam search decoder is a heuristic search that limits its exploration space to its beam size. Unlike the phrase-based decoder, it does not guarantee finding the optimal solution. It visits each state and keeps only the $K$ best states in a priority queue. Only the $K$ best states in the queue are expanded to subsequence $K$ best states (from left to right); therefore, not all possible states are considered– only the promising ones. The computational complexity of the beam search does not grow with the value of the Markov order. For all cases, the complexity of decoding is $O(K*P*n)$, where $K$ is the beam size, $P$ is phoneme options, and $n$ is grapheme length. In the exact algorithm, the complexity becomes $O(P^{M+1}*n)$, where $M$ is the value of the Markov order. Using the beam search in the model can be considered a trade-off between search accuracy and the capability of the model to explore in a higher-order Markov space. The value of $K$ in the beam search controls the size of the search history; this impacts the solution quality. This value is difficult to predict optimally. In practice, the value of $K$ is set using a development set. Figure 4.8 shows the system performance in terms of the word accuracy as a function of the beam size on a development set. The performance starts to converge quickly and shows no further improvement for values grater than $20$. At a beam size equal to 20, the system obtains $69.07\%$ word accuracy which is slightly lower than the word accuracy achieved by the exact algorithm at $69.13\%$.

Figure 4.9 shows the word accuracy with different values of $n$ in the online discriminative system that includes *only* the joint $n$-gram features. The accuracy reaches a maximum for $n$ = 4, and actually falls off for larger values of $n$. This is likely caused by the model using its expanded expressive power to memorize sequences of operations, overfitting the training data. Such overfitting is less likely to happen in the generative joint $n$-gram model,

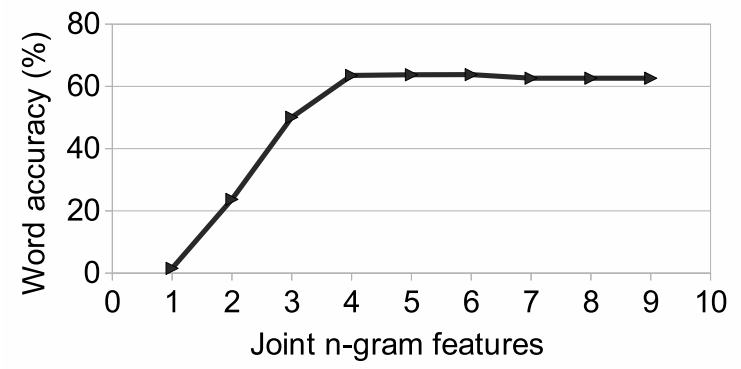


Figure 4.9: System accuracy as a function of $n$-gram size.

| Data set | Training size | DirecTL+ | DirecTL | Joint $n$-gram |
|---|---|---|---|---|
| Celex | 40K | **89.23** | 88.54 | 88.58 |
| CMUdict | 106K | **76.41** | 75.41 | 75.47 |
| OALD | 57K | **85.54** | 82.43 | 82.51 |
| NETtalk | 19K | **73.52** | 70.18 | 69.00 |
| Brulex | 25K | **95.21** | 95.03 | 93.75 |

Table 4.10: Grapheme-to-phoneme conversion accuracy.

which smooths higher-order estimates very carefully.

To directly compare results with the generative joint $n$-gram model of Bisani and Ney [2008], I follow exactly the same data splits as used in their publication. The data sets include English Celex, NETtalk, OALD, CMUdict, and French Brulex. The training sizes range from 19K to 106K words.

Table 4.10 shows the performance of the DirecTL and DirecTL+ systems in comparison with the joint $n$-gram approach. DirecTL refers to the system described in Section 4.2. It includes only context, transition and linear-chain features as in Table 4.4. DirecTL+ is the DirecTL system that includes the joint $n$-gram features as presented in Table 4.9, in addition to the existing feature sets. The DirecTL+ model uses a beam search with a beam size of 50, joint 6-gram features ($5^{th}$ Markov order). Other parameters are set on the respective development sets. The full DirecTL+ system outperforms both DirecTL and the joint $n$-gram model on all data sets. DirecTL+ improves system performance over DirecTL by adding the joint $n$-gram features. The relative error deductions ranged from 3.6% to 17.3%, establishing a new state-of-the-art for the task.

## 4.6 Summary

I presented grapheme chunking model that incorporates many-to-many alignments into existing classification-based G2P conversion systems. The chunking model aims to dynamically discover double graphemes in the testing words, so that they are pre-processed and segmented in accordance with the alignments in the training data. Applying the chunking model and many-to-many alignments to the instance-based learning classifier enables improvements ranging from 2.7% to 7.6% over the baseline system in absolute word accuracies. To further improve the classification performance, a language model is applied as a post-processing step to guide the model to natural output sequences. Improvements of 1% to 2% in absolute word accuracies are achieved. These improvements illustrate the power of models that consider G2P conversion as a sequence prediction problem.

Although the proposed approach shows improvement over classification-based approaches in the literature, it suffers from disjointly modeling each component. Like other pipeline systems, errors made in earlier components are forwarded to the next stage. It is unlikely that later processes can recover or even be aware of these earlier errors. The joint processing approach, DirecTL, is proposed to collapse a pipeline process. By unifying all the models in this pipeline process prevents propagation of errors. The chunking and language models are collapsed into one single model via a phrase-based decoder. The decoder simultaneously searches for the best grapheme segments and the most likely sequence of phonemes. The language model is implicitly included as features information. The features include wide context source $n$-grams, output transition and linear-chain features. The system trains feature weights using the max-margin online discriminative algorithm based on MIRA [Crammer and Singer, 2003]. At the time, DirecTL surpassed all state-of-the-art results on the CELEX, Nettalk, CMUDict and Brulex data sets. Its improvements ranged from 2% to 8% in absolute word accuracies.

Lexical and phonemic stress markers were also incorporated into the DirecTL system to further improve the grapheme generation performance and to generate more complete outputs for text-to-speech systems. Adding lexical stress improves overall performance by directly providing extra stress information on the graphemes, so the system is aware of stress information during generation. The experiments demonstrate better results for the proposed system than the widely used speech synthesis system, Festival.

In addition, an algorithm for training without alignments in DirecTL is presented to explore an alternative paradigm. Grapheme-to-phoneme alignments can be viewed as latent

variables in the learning model. The training algorithm is composed of two state training components: (1) finding the best hidden structure given the current feature weights, and (2) updating the feature weights using the current best hidden structures. Unlike results reported in other domains, here the latent alignment model does not improve over the DirecTL system that trains with fixed hidden structures found by m2m-aligner.

Finally, DirecTL+ is proposed as an extension to DirecTL. DirecTL+ combines the well-known joint $n$-gram model for G2P with the DirecTL system. The system incorporates the joint $n$-gram information as new features and applies a beam search in place of the exact algorithm. Empirical results show that the combined system outperforms both DirecTL and the joint $n$-gram model of Bisani and Ney [2008], establishing a new state-of-the-art for G2P conversion.

# Chapter 5

# Transliteration

Transliteration plays an important role in many natural language processing systems, especially in Machine Translation (MT) and Cross Language Information Retrieval (CLIR). Unlike translation, transliteration is the task of converting a given name from one (source) language to a phonetically equivalent name in another (target) language. An exact phonetic equivalence between two languages may not be possible due to differences in their phoneme sets. Approximate phonetic equivalences are generally acceptable as transliterations. These desired transliteration is usually based on human intuition as to the equivalent of a source language name in the target language [Li et al., 2009]. It is a challenging task because the input and output languages use different writing and sound systems. The problem requires even more precision when one transliterates a language name written in another language back to its original language, which is called "back-transliteration". The evaluation of the back-transliteration task is usually less forgiving because each input often refers to only one exact spelling in the original language.

Many techniques have been proposed for transliteration and back-transliteration [Klementiev and Roth, 2006, Knight and Graehl, 1998, Li et al., 2004, Sproat et al., 2006, Zelenko and Aone, 2006]. Recently, these problems have received a lot of attention in the NLP community resulting in two consecutive years of shared tasks on name transliteration at the Annual meeting of the association for Computational Linguistics (ACL) conferences [Kumaran et al., 2010, Li et al., 2009, 2010]. These shared tasks can be categorized into two different tasks: (1) name transliteration generation, and (2) name transliteration mining. Name transliteration generation task focuses on training a system to generate a target language names given a source language name, while a system in the mining task identifies a list of transliteration pairs given parallel text written in both languages.

In this chapter, I apply the DirecTL approach (Chapter 4) to name transliteration, specif-

ically to the NEWS 2009 and 2010 shared tasks. DirecTL-based approaches for name transliteration generation are described in Section 5.1. This is joint work with Aditya Bhargava, Qing Dou, Kenneth Dwyer, Mi-Young Kim and Grzegorz Kondrak, published in [Jiampojamarn et al., 2009, 2010b]. The language-specific approaches for name transliteration presented in Section 5.1.3 were mainly contributed by Aditya Bhargava, Qing Dou, Mi-Young Kim and Grzegorz Kondrak. These approaches enhance the DirecTL framework by incorporating language-specific knowledge. I present name transliteration mining in Section 5.2. This is joint work with Shane Bergsma, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak published in [Jiampojamarn et al., 2010b]. Five main systems participated to the shared task from the University of Alberta. The DirecTL systems were applied as a generation-based approach to the mining task. The key idea is to mine transliteration pairs based on how similar candidate words are to generated transliterations by DirecTL. The NED, Bergsma and Kondrak [2007], StringKernel and StringMatch approaches were contributed by Grzegorz Kondrak, Shane Bergsma, Kenneth Dwyer, and Qing Dou respectively. StringMatch is an improved approach for extracting Chinese candidate words when using DirecTL+ as the transliteration generation model. I provide the summary of this chapter in Section 5.3.

## 5.1 Transliteration generation

In principle, the task of name transliteration generation can be viewed as a similar task to grapheme-phoneme conversion. A system learns from examples of source-target transliteration pairs in training data and then generates target language names given source names at the test time. Unlike G2P, both the inputs and outputs of name transliteration systems are represented in graphemes. The pronunciation of these graphemes, i.e. their corresponding phonemes, are highly correlated. Previously, Knight and Graehl [1998] proposed a four-stage cascade of finite-state transducers that connect graphemes of two languages with their phoneme mappings. They developed an English-Japanese transliteration system consisting of four models: (1) an English word model, (2) an English pronunciation model, (3) an English-Japanese sound conversion model, and (4) a Japanese spelling model. This system requires not only pronunciation dictionaries for both languages to train model 2 and model 4 but also, more importantly, example pairs of English and Japanese sounds to train model 3. A more attractive approach is to generate transliteration directly from graphemes without converting to to phonemes at all. The joint source-channel model of [Li et al., 2004] is

trained to directly transduce graphemes of the source language to the target language. The approach is similar to the joint $n$-gram model in G2P [Bisani and Ney, 2008]. Although the joint source-channel model was originally proposed for English-Chinese name transliteration, it is promising to apply the model to other language pairs due to the fact that it requires only a set of transliteration examples without any other parallel data or dictionaries.

In the context of the Machine Transliteration Shared Tasks in the Named Entities Workshops, NEWS 2009 [Li et al., 2009] and NEWS 2010 [Li et al., 2010], the task is to develop a machine transliteration system for one or more language pairs. A provided training set consists of example pairs of transliteration names in source and target languages. In some cases, there is more than one acceptable name given a source language name. These multiple correct answers are due to spelling variations. Results that are obtained from a system that trains on only the provided data are referred to as "standard" results. These standard results ensure a meaningful performance comparison between the various proposed methods. "Non-standard" results refer to results achieved by a system that uses other data, including additional example pairs or linguistic resources.

Among the standard systems, in the transliteration generation task, phrase-based statistical machine translation [Koehn et al., 2003] was one of the most widely-used techniques [Finch and Sumita, 2009, 2010, Noeman, 2009, Rama and Gali, 2009, Song et al., 2009, 2010]. Statistical machine translation is applied by simply trading words in translation for characters in transliteration. Other systems [Aramaki and Abekawa, 2009, Shishtla et al., 2009] adopted Conditional Random Fields (CRFs) [Lafferty et al., 2001], formulating the transliteration problem as a sequence labeling problem. Word alignment models from machine translation, especially GIZA++ [Och and Ney, 2003] were commonly used in these systems to find character alignments between the source and target training names. The pair $n$-gram approach of Jansche and Sproat [2009] shares the same principle as the joint $n$-gram model proposed in grapheme-to-phoneme conversion [Bisani and Ney, 2002, 2008] and the joint source-channel model of Li et al. [2004]. As in other shared tasks, combinations of several different models via re-ranking yield good performance [Das et al., 2009, 2010, Finch and Sumita, 2010, Oh et al., 2009, Song et al., 2009].

Additional resources such as name dictionaries, pronunciation dictionaries, and additional training data from Linguistic Data Consortium, WWW search, and Wikipedia search were applied in [Hong et al., 2009, Jansche and Sproat, 2009]. Among these resources, additional lists of names extracted from WWW search and Wikipedia greatly improve transliteration performance to as much as 90% word accuracy over a standard system that achieves

| Data set | Data source | Training | Development | Testing |
|---|---|---|---|---|
| EnHi | Microsoft Research India | 9,975 | 974 | 1,000 |
| EnTa | Microsoft Research India | 7,974 | 987 | 1,000 |
| EnKa | Microsoft Research India | 7,990 | 968 | 1,000 |
| EnRu | Microsoft Research India | 5,977 | 943 | 1,000 |
| EnCh | Institute of Infocomm Research | 31,961 | 2,896 | 2,896 |
| EnKo | CJK Institute | 4,785 | 987 | 989 |
| EnJa | CJK Institute | 23,225 | 1,492 | 1,489 |
| JnJk | CJK Institute | 6,785 | 1,500 | 1,500 |

Table 5.1: Evaluation data sets used in NEWS 2009.

| Data set | Data source | Training | Development | Testing |
|---|---|---|---|---|
| ArAe | CJK Institute | 25K | 2.5K | 2.5K |
| EnBa | Microsoft Research India | 10K | 2K | 2K |
| EnTh | NECTEC | 26K | 2K | 2K |
| ThEn | NECTEC | 24K | 2K | 2K |
| ChEn | Institute of Infocomm Research | 25K | 5K | 2K |

Table 5.2: Additional evaluation data sets used in NEWS 2010.

60% word accuracy for English-Russian and English-Chinese language pairs [Jansche and Sproat, 2009].

There were eight data sets used in the NEWS 2009 evaluation, from three different sources: (1) Microsoft Research India, (2) Institute of Infocomm Research, and (3) CJK Institute. The data sets are English-Hindi (EnHi), English-Tamil (EnTa), English-Kannada (EnKa), English-Russian (EnRu), English-Chinese (EnCh), English-Korean (EnKo), English-Japanese Katakana (EnJa) and Japanese Romaji to Japanese Kanji (JnJk). The training sets consist of from 5K to 32K examples; the development and test sets consist of from 1K to 3K names. Table 5.1 summarizes the data sets in the NEWS 2009 shared task. The shared task makes no distinction between forward and backward transliteration; however, it should be noted that the EnCh, EnKo, and EnJa data sets contain Western names (forward transliteration), while the JnJk data set contains only native Japanese names (backward transliteration). EnHi, EnTa and EnKa contain Indian and Western names of mixed origin.

In the NEWS 2010 shared task, there are 12 data sets including 7 data sets (EnHi, EnTa, EnKa, EnCh, EnKo, EnJa and JnJk) from NEWS 2009 and 5 new data sets: Arabic-English (ArAe), English-Bangla (EnBa), English-Thai (EnTh), Thai-English (ThEn) and Chinese-English (ChEn). Table 5.2 shows these additional five data sets. ChEn, EnTh and ThEn contain Western names, while ArAe consists of Arabic origin names. Each of these sets is thus exclusively either forward or backward transliteration. Like other Indian data sets

(EnHi, EnTa, EnKa), EnBa contains a mixed of Indian and Western names. For the data sets from NEWS 2009, the training data contain both the training and development sets from the previous year. The 2010 development sets are the 2009 test sets and the new test sets are constructed from totally different sources. It is interesting to note that while all NEWS 2009 evaluation sets were randomly split to create training, development and testing sets, the NEWS 2010 testing data are not necessarily from the same domain as the training and development sets. The tasks have thus become more difficult but more realistic than the previous year.

Each system is asked to provide a ranked list of up to 10 candidate answers for each test name. Since a name may have multiple correct transliterations, especially in the forward transliteration task, all reference answers are treated equally in the evaluation. The transliteration results are evaluated using 6 different evaluation metrics including:

1. Word accuracy in top-1 (ACC) measures the correctness of the top-1 answer (the top ranked answer in the list). The answer is considered to be correct if it completely matches one of the transliteration names in the reference list. Equation 5.1 shows the ACC calculation, where $N$ is the total number of names in the test set, $r_{i,j}$ is the $j$-th reference transliteration for the $i$-th testing name, $c_{i,k}$ is the $k$-th candidate transliteration for the $i$-th testing name.

$$ACC = \frac{1}{N}\sum_{i=i}^{N}\begin{cases} 1 & \text{if } \exists r_{i,j} : r_{i,j} = c_{i,1} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

2. Fuzziness in top-1 (Mean F-score) measures how different, on average, the top transliteration candidate is from its closest true transliteration in the reference list. The F-score measurement is a function of precision and recall, calculated based on the Longest Common Subsequence (LCS) between a candidate ($c$) and a reference ($r$):

$$LCS(c,r) = \frac{1}{2}(|c| + |r| - ED(c,r)) \tag{5.2}$$

where, $ED(c,r)$ is the minimum edit distance between $c$ and $r$, $|c|$ and $|r|$ are the numbers of Unicode characters in string $c$ and $r$, respectively. The minimum edit distance function uses an equal cost of adding, removing and replacing a character. The recall, precision and F-score for transliteration is evaluated using the following

equations:

$$R = \frac{1}{N}\sum_{i=1}^{N}\frac{LCS(c_{i,1}, r_{i,m})}{|r_{i,m}|} \quad (5.3)$$

$$P = \frac{1}{N}\sum_{i=1}^{N}\frac{LCS(c_{i,1}, r_{i,m})}{|c_{i,m}|} \quad (5.4)$$

$$F = 2\frac{R * P}{R + P} \quad (5.5)$$

where $r_{i,m}$ is the reference that $c_{i,i}$ matches best in the reference list for the $i$-th testing name. Since each transliteration name in the reference list is considered equally correct, a candidate transliteration is evaluated based on the reference answer that gives the lowest minimum edit distance.

3. Mean Reciprocal Rank (MRR) measures the system's $n$-best candidate answers. $\frac{1}{MRR}$ is roughly the average rank of the correct transliteration in the $n$-best list. Like in ACC, a candidate answer is correct if it matches any transliteration in the reference list. An MRR close to 1 indicates that the correct answer is usually at the top of the $n$-best list.

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} \min_k \frac{1}{k} & \exists r_{i,j}, c_{i,k} : r_{i,j} = c_{i,k} \\ 0 & \text{otherwise} \end{cases} \quad (5.6)$$

4. $\text{MAP}_{\text{ref}}$ measures the precision of the $n$-best candidate answers. If a system generates all $n_i$ correct transliteration answers for the $i$-th source name in its top $n_i$-list, then it receives a perfect MAP=1. The $\text{MAP}_{\text{ref}}$ measurement is defined as:

$$MAP_{\text{ref}} = \frac{1}{N}\sum_{i=1}^{N}\frac{num(i, n_i)}{n_i} \quad (5.7)$$

where $num(i, n_i)$ is the number of correct transliterations for the $i$-the source name in the $n_i$-best list, and $n_i$ is the number of transliteration variations in the reference list.

5. $\text{MAP}_{10}$ is similar to $\text{MAP}_{\text{ref}}$ but it is computed with respect to a fixed size (10-best) list of candidates instead of using the number of correct transliteration names in the reference ($n_i$).

6. $\text{MAP}_{\text{sys}}$ is similar to $\text{MAP}_{\text{ref}}$ but it is computed with respect to the size of candidate list proposed by the system, $K_i$, instead of using $n_i$ or a fixed number.

### 5.1.1 Approaches to transliteration generation

Name transliteration generation is an example of a string transduction application. It is closely related to grapheme-to-phoneme conversion by both the structure of the problem and the phonetic equivalence between source and target language names. In general, the approaches proposed for transliteration generation are based on many-to-many alignments (Chapter 3) and the generation models (Chapter 4). In the transliteration task, training data consist of word pairs that map source language words to words in the target language. The matching between character substrings in the source word and target word is not explicitly provided. These hidden relationships are generally known as "transliteration alignments". It is the same situation as in the grapheme-phoneme alignment in G2P conversion. The m2m-aligner can therefore be used to find the alignments in the training data.

Both DirecTL and DirecTL+ were applied to the NEWS 2009 and 2010 shared tasks [Jiampojamarn et al., 2009, 2010b]. As in G2P, these systems train on many-to-many alignments via the online discriminative training framework using the margin-based method, MIRA [Crammer and Singer, 2003]. A phrase-based decoder [Zens and Ney, 2004] is applied to automatically find the best segmentations that generate the most likely output sequence. The transliteration models use the same feature templates as presented in Tables 4.4 and 4.9. Although most of the components of the G2P system can be directly applied to name transliteration, one major difference is how to efficiently train the model with multiple target language names for each source name. This is equivalent to having pronunciation variations in grapheme-to-phoneme conversion. Pronunciation variations were previously ignored during training. The problem of multiple target language names is closely related to multi-labeled classification problems [Elisseeff and Weston, 2001, Zhou and Zhang, 2006]. Section 5.1.2 describes the training method for DirecTL when there is more than one correct output in the training data. The proposed method is based on a loss function, over the correct outputs and the predicted answers at each iteration.

Another key difference from G2P is the need for language-specific enhancements (Section 5.1.3). These modifications are inspired by linguistic knowledge including an intermediate phonetic representation for Chinese, Korean, and Japanese, and language origin detection for Indian-related languages. Performance comparisons using these language-specific approaches, as well as the official results of the NEWS shared tasks, are presented in Section 5.1.4.

### 5.1.2 Training with multiple answers

One difficulty when DirecTL models are trained on the transliteration training data is that each source language name may have multiple correct target language names. Unlike the DirecTL models, the generative methods [Bisani and Ney, 2008, Jansche and Sproat, 2009] have no difficulty to leverage multiple references in the probability models. In the DirecTL, the complication is in the update stage. Equation 5.8 restates the update method, where $\alpha_n$ is the updated weights, $\alpha_o$ is the current weights, $\hat{T_n}$ is the $n$-best system outputs, $\Phi(s,t)$ is a set of features generated from the source language name $s$ and target language name $t$, and $\ell(t,\hat{t})$ is the loss function between the correct output $t$ and proposed output $\hat{t}$.

$$\begin{array}{l} \min_{\alpha_n} \parallel \alpha_n - \alpha_o \parallel \\ \text{subject to } \forall \hat{t} \in \hat{T_n} : \\ \alpha_n \cdot (\Phi(s,t) - \Phi(s,\hat{t})) \geq \ell(t,\hat{t}) \end{array} \tag{5.8}$$

Previously, there was only one correct output sequence $t$ for each input $s$. The constraints in the equation essentially consist of a list of features that differ between the correct and incorrect outputs. Since it is possible in name transliteration to have more than one correct target language name, one simple solution is to randomly choose a target language name beforehand and ignore the rest. This solution simplifies the complexity of the problem but sacrifices other potentially-useful ground truth information that the model could learn from.

Creating constraints for all pairs between $\hat{t} \in \hat{T_n}$ and $t \in T$;where $T$ is the list of correct transliterations, results in an unstable model that is unlikely to converge. Consider an extreme case where the model produces an almost correct answer $\hat{t}$ which is close to one target in the reference list, $t_1$. The constraints consist of not only the differences between $\Phi(s,t_1)$ and $\Phi(s,\hat{t})$, but also other constraints for other $t \in T_n, t \neq t_1$. Such essentially unrelated constraints may be hard to satisfy simultaneously. In addition, the number of constraints grows in the proportion to the Cartesian product $|T_n|\chi|\hat{T_n}|$, instead of the size of the $n$-best list.

To incorporate all correct transliteration names into the learning model, I proposed a loss-based selection method which dynamically choose an appropriate correct transliteration name for each generated output. The model selects a correct transliteration $t$ from the reference list $T$ such that it has the minimum loss to the candidate $\hat{t}$. This loss criterion is highly correlated to the evaluation matrices. When there are multiple references, these metrics reward a candidate that matches one of the transliteration names or has minimal edit distance. The final update formulation is presented in Equation 5.9. The weights of

| Strategy | Top-1 word accuracy |
|---|---|
| Baseline | 45.76 |
| max score | 45.87 |
| min score | 47.14 |
| max loss | 46.81 |
| min loss | 47.69 |

Table 5.3: Top-1 word accuracy performance of different training strategies on the EnJa development set.

each proposed candidate in the $n$-best list are updated toward the closest correct answer $t$ for that candidate. The size of the constraints in the equation is in proportion to the size of the $n$-best list.

$$\begin{aligned} &\min_{\alpha_n} \parallel \alpha_n - \alpha_o \parallel \\ &\text{subject to } \forall \hat{t} \in \hat{T_n} : t = \arg\min_{t' \in T} \ell(t', \hat{t}) \\ &\alpha_n \cdot (\Phi(s,t) - \Phi(s,\hat{t})) \geq \ell(t,\hat{t}) \end{aligned} \tag{5.9}$$

Alternatively, one can choose the correct answer, $t$, that provides the largest loss instead of the minimum one, so that the training model can train on more difficult references in term of the loss function. However, this training procedure is unstable as shown in the following situation. Let's assume that there are two correct answers, $t_1, t_2 \in T$, and, first, the training model prefers the prediction output, $\hat{t}$, which is close to the reference answer $t_1$ but the training model updates the feature weights $\alpha$ toward the reference $t_2$ as it has a higher loss. Later, after the update, the model would prefer to generate an output $\hat{t}$ that is close to the reference answer $t_2$ and again it is forced to update the feature weights toward the reference $t_1$. These alternate updates between reference $t_1$ and $t_2$ lead to an unstable model during training for each example.

Another possible solution is to choose references based on their model scores, $\alpha \cdot \Phi(s,t)$. Selecting the reference that has the minimum model score guarantees to update the feature weights to separate incorrect outputs, $\hat{t}$ from the lowest bound of the correct outputs, $t = \arg\min_{t' \in T} \alpha \cdot \Phi(s,t')$. Similarly, this strategy can cause the alternate updates between two references. For example, if a majority of training examples suggests the model to favor the reference $t_1$ in the example. At the update, the feature weights are forced to update toward the reference $t_2$ since it has a lower model score than the reference $t_1$. Therefore, the alternate updates between two choices can occur over the overall training examples when the feature weights are updated in favor of the reference $t_1$ on other examples in the training set.

Table 5.3 shows top-1 word accuracies of the EnJa development set using different training strategies discussed in this section. "Baseline" is a baseline system that selects

each target reference at random. The "max score" and "min score" strategies select each target reference based on the maximum and minimum model scores respectively. Similarly, the "max loss" and "min loss" strategies are based on the loss function. All training strategies which incorporate multiple references improve word accuracy performance over the baseline system. The min score strategy outperforms the max score criteria because it guarantees to update feature weights on the largest difference of the feature values, $\Phi(s,t) - \Phi(s,\hat{t})$. The min loss strategy obtains a better performance than the max loss model because it avoids the alternate updates of multiple references. The min loss model is slightly better than the min score model since it updates the model toward the closest reference. It is less aggressive than the min score model that always updates with the largest value of the feature differences.

### 5.1.3 Language-specific approaches to name transliteration

The DirecTL framework for name transliteration is a language independent approach. The system learns to directly generate target language names without requiring specific language knowledge. It transliterates substrings of characters from a source language to a target language regardless of whether the languages uses alphabetic, syllabic, or ideographic scripts. Although the many-to-many alignment approach has the ability to find small units of substrings that are sufficiently represent the scripts of source and target languages, it is interesting to pre-process the non-alphabetic languages to be closer in form to an alphabetic script. For the NEWS 2009 and NEWS 2010 shared tasks, we propose intermediate representations for name transliteration in Chinese, Korean and Japanese. These three language using non-alphabetic writing system.

Chinese Pinyin is the most common romanization system for Chinese Mandarin. It uses the 26 letters of English alphabet to represent the sounds of Chinese Mandarin. Representing each Chinese character with Pinyin characters can help the model find better substring alignments between Pinyin and English alphabets. Instead of finding English substrings that align with individual Chinese characters, the model can generalize the mappings between Pinyin symbols and the English alphabet. This pre-processing approach significantly reduces the number of distinct symbols in the target side: from 370 different Chinese characters to 26 Pinyin symbols. The Chinese Pinyin pre-processing steps are only applied to the alignment model for better generated alignments. The pre-processing steps for the English-Chinese transliteration task are as follows: (1) Chinese characters are represented in their

corresponding Pinyin form via a standard conversion table, (2) Many-to-many alignments are constructed on the English-Pinyin training data, (3) Pinyin characters are converted back to their corresponding Chinese characters, (4) finally, the aligned English-Chinese data are used to train the generation model.

Alternatively, one can use the Pinyin representation for both alignment and generation models. Such a system learns to generate sequences of Pinyin given English names. Separate post-processing steps are required to transfer the Pinyin sequences back to the corresponding Chinese characters. However, converting back the imperfect sequences is not a trivial task. One Chinese character usually corresponds to a combination of 3-4 Pinyin symbols. To avoid further conversion errors, the Chinese Pinyin conversion is only applied during the alignment process, during the training, where the reference Chinese names are available.

Unlike Chinese, Korean characters can be decomposed into two or three components called "Jaso": an initial consonant, a middle vowel and optionally a final consonant. For the English-Korean transliteration task, it is important to decompose the Korean symbols into their corresponding Jaso. The conversion between Korean characters and their Jaso components is a lossless process via a conversion table. Instead of training a transliteration model on the original format, English-Jaso data are used to train the transliteration model. After transliteration, the Jaso output is then converted back to Korean characters. It is possible that a generated Jaso sequence can not be converted back to Korean because of system errors. However, simple correction rules are sufficient to convert the illegal Jaso sequences to legal ones. The rules include (1) replacing two consecutive vowels with a complex vowel, (2) inserting silent consonant *o* (i-eung) between two vowels, (3) inserting a vowel - (eu) between two consonants or, for three consecutive consonants, placing the vowel in the most probable position according to the training data. The system removes any illegal Jaso sequence from its $n$-best transliteration outputs that can not be recovered.

Although Japanese Katakana is often used for transcription of words from foreign languages, replacing each Katakana symbol with one or two English letters via a standard romanization table helps in the alignments. The transliteration model is trained to generate names written in romanized form. These forms are then converted back to their original Japanese Katakana. Unlike Korean, most Katakana symbols are represented by single vowels or consonant-vowel pairs. The only apparent ambiguity involves the letter *n*, which can either stand by itself or cluster with the following vowel letter. The system resolves this ambiguity by always assuming the latter, unless the letter *n* occurs at the end of the word.

Only minority illegal sequences are generated by the transliteration system in its $n$-best outputs. The system takes a simple solution: it removes any sequence from the output list that it is unable to convert back to Japanese Katakana.

Unlike Chinese, Korean and Japanese, Russian is an alphabetic writing system. It uses a Cyrillic alphabet that is largely phonetic. It is relatively straightforward to convert the Cyrillic script to the IPA representation. A conversion table is used for mapping between Cyrillic and the IPA representation. There is no ambiguity in the backward or forward conversion. This conversion is mostly for accommodating the ALINE algorithm that requires both source and target names in the IPA representation (Section 3.2).

All transliteration data for Indian languages in the NEWS shared tasks consist of both Western and Indian origin names. Therefore, the provided training data is a combination of both forward and backward transliterations. This is a great opportunity to explore whether language-origin identification can result in better transliterations. The idea is to separate the training data, and train one model for forward transliteration and another for backward transliteration. We apply the language identification model of Bhargava and Kondrak [2010]. The system is based on support vector machines (SVMs); classification is trained on a small number of examples manually tagged as being Indian or non-Indian in origin. Instead of splitting the data into two disjoint sets, the identification model generate scores for being a name of Indian or non-Indian origin. Then, a threshold value determines if the given name should be excluded from the set. As a result, only the most likely Indian names are excluded from the non-Indian set and vice versa. During testing, each transliteration model generates a list of target transliterations given a source name. The two lists are combined using a linear combination over the mean reciprocal ranks. The linear combination weights are taken from the scores of the identification model for the source name, so that the final results favor the corresponding model.

### 5.1.4 Summary of evaluation and results

The approaches are evaluated using the NEWS 2009 and NEWS 2010 name transliteration generation tasks. The DirecTL system was mainly applied to the NEWS 2009 task and the DirecTL+ system for the NEWS 2010 task the following year. The many-to-many alignments were constructed by using two alignment approaches: (1) the m2m-aligner, and (2) ALINE (Chapter 3). ALINE requires the IPA representations for both source and target names. This limits the algorithm to some specific languages; for example, English, Russian, Chinese Pinyin, and Japanese Katakana can be represented in the Latin alphabet. Unlike

ALINE, the m2m-aligner produces alignments regardless of the written scripts used in the languages.

For all the generation tasks, the provided data are preprocessed as follows:

1. converting all characters in the source word to lower case.

2. removing non-alphabetic characters unless they appear in both the source and target words.

3. normalizing whitespace surrounding a comma, so that there are no spaces before the comma and exactly one space following the comma.

4. separating multi-word name pairs into separate single-word name pair, using whitespace as the separator and assuming a monotonic matching. Any example that has a different number of words in the source and target is discarded from the training set.

In the ArAe task, there are cases where an extra space is added to the target names when transliterating from Arabic to English; for example, “Al Riyard”, and “El Sayed”. In order to prevent the pre-processing from removing too many examples, unequal matching is allowed if the source name is a single word.

During testing, the pre-processing steps are applied in the same manner. Transliteration for multi-word names are generated from the $n$-best answers of the single words by ranking the combined scores that make up the test words.

Table 5.4 shows the results of the DirecTL system on the NEWS 2009 transliteration generation tasks for English-Chinese (EnCh), English-Hindi (EnHi), English-Japanese Katakana (EnJa), English-Korean (EnKo), English-Russian (EnRu), and Japanese Romaji-Japanese Kanji (JnJk). “+INT(m2m)” refers to using the language-specific approaches presented in Section 5.1.3 with the m2m-aligner, while “+INT(ALINE)” refers to the same but with the ALINE algorithm generating the alignments. “Combined” is a system combining all available answers using a simple voting method. “+MC” in EnHi refers to using a manual cleaning step performed by a Hindi speaker to fix about 43 transliteration pairs that have a disagreement between the number of source and target words.

For all experiments, the system intentionally produces 10-best outputs without a mechanism to filter low-quality target names. Therefore, the $\text{MAP}_{10}$ and $\text{MAP}_{sys}$ results are the same. The language-specific approaches significantly improve the results for the EnCh task but not for EnJa and EnRu. The results suggest that using an intermediate representation

| Data set | System | ACC | F-score | MRR | $MAP_{ref}$ | $MAP_{10}$ | $MAP_{sys}$ |
|---|---|---|---|---|---|---|---|
| EnCh | DirecTL | 0.717 | 0.890 | 0.785 | 0.717 | 0.237 | 0.237 |
| | +INT(m2m) | 0.734 | 0.895 | 0.807 | 0.734 | 0.244 | 0.244 |
| | +INT(ALINE) | 0.732 | 0.895 | 0.803 | 0.732 | 0.242 | 0.242 |
| | Combined | 0.746 | 0.900 | 0.814 | 0.746 | 0.245 | 0.245 |
| EnHi | DirecTL | 0.498 | 0.890 | 0.603 | 0.488 | 0.195 | 0.195 |
| | +MC | 0.509 | 0.893 | 0.610 | 0.498 | 0.198 | 0.198 |
| EnJa | DirecTL | 0.500 | 0.847 | 0.604 | 0.487 | 0.199 | 0.199 |
| | +INT(m2m) | 0.492 | 0.843 | 0.597 | 0.478 | 0.198 | 0.198 |
| | +INT(ALINE) | 0.510 | 0.848 | 0.614 | 0.496 | 0.202 | 0.202 |
| | Combined | 0.505 | 0.850 | 0.616 | 0.493 | 0.204 | 0.204 |
| EnKo | DirecTL | 0.387 | 0.693 | 0.469 | 0.387 | 0.146 | 0.146 |
| EnRu | DirecTL | 0.613 | 0.928 | 0.696 | 0.613 | 0.212 | 0.212 |
| | +INT(m2m) | 0.608 | 0.927 | 0.694 | 0.608 | 0.212 | 0.212 |
| | +INT(ALINE) | 0.607 | 0.927 | 0.690 | 0.607 | 0.211 | 0.211 |
| | Combined | 0.608 | 0.927 | 0.693 | 0.608 | 0.211 | 0.211 |
| JnJk | DirecTL | 0.560 | 0.847 | 0.604 | 0.487 | 0.199 | 0.199 |

Table 5.4: Evaluation results on NEWS 2009 transliteration generation.

for Russian in fact degrades overall performance. EnJa results are improved from the baseline DirecTL only when ALINE is used to generate alignments. The EnCh results illustrate the advantage of using Pinyin representation. A large improvement of 3% absolute ACC is achieved by representing Chinese symbols with Pinyin.

The EnHi results with manual cleaning method (EnHi+MC) give an indication of the potential of DirecTL when it is trained on less-noisy data. The difference in ACC with cleaning can be as much as 1% in absolute ACC.

Comparing the best results in the table to other systems reported in the NEWS 2009 transliteration shared task [Li et al., 2009], the DirecTL approach obtains the best results in the EnCh, EnHi and EnRu tasks among 31 participating systems. Including the joint $n$-gram features in DirecTL+ further improves performance to 0.742, 0.503 and 0.618 ACC on the EnCh, EnHi and EnRu tasks, respectively, versus 0.717, 0.498 and 0.613 with DirecTL.

Table 5.5 shows the DirecTL+ results on the NEWS 2010 transliteration generation task. The overall performance except for EnKo drops from the NEWS 2009 results. This performance drop could be because the test sets in NEWS 2010 come from totally different sources than the 2010 training and development sets. In NEWS 2009, all training, development, and test sets come from the same sources. Comparing to other systems participating in the shared tasks, DirecTL+ is ranked either the first or second on all data sets.

Using the Pinyin representation for Chinese gives improvements over the baseline system, but in smaller amounts compared to the NEWS 2009 results. "+LangID" refers to the

| Data set | ACC | F-score | MRR | $MAP_{ref}$ |
|---|---|---|---|---|
| EnCh | 0.357 | 0.703 | 0.419 | 0.342 |
| +INT(m2m) | 0.360 | 0.707 | 0.429 | 0.345 |
| +INT(ALINE) | 0.362 | 0.704 | 0.429 | 0.348 |
| Combine | 0.363 | 0.707 | 0.430 | 0.348 |
| ChEn | 0.137 | 0.740 | 0.198 | 0.137 |
| EnTh | 0.378 | 0.867 | 0.467 | 0.378 |
| ThEn | 0.352 | 0.861 | 0.450 | 0.352 |
| EnHi | 0.456 | 0.884 | 0.559 | 0.456 |
| +LangID | 0.456 | 0.885 | 0.558 | 0.456 |
| EnTa | 0.390 | 0.891 | 0.512 | 0.390 |
| EnKa | 0.341 | 0.867 | 0.460 | 0.341 |
| EnJa+INT(m2m) | 0.398 | 0.791 | 0.507 | 0.398 |
| EnKo+INT(m2m) | 0.554 | 0.770 | 0.672 | 0.554 |
| JnJk | 0.126 | 0.426 | 0.201 | 0.127 |
| ArAe | 0.464 | 0.924 | 0.535 | 0.265 |
| EnBa | 0.395 | 0.877 | 0.512 | 0.395 |

Table 5.5: DirecTL+ results on the NEWS 2010 transliteration generation tasks.

system that applies the language identification model for the EnHi task. The system obtains exactly the same ACC performance as the fully language-independent DirecTL+ approach. On the other hand, the Korean Jaso approach greatly improves overall performance, by improving as much as 17% in absolute ACC compared to the language independent approach.

## 5.2 Transliteration mining

Most of the data for the NEWS transliteration generation tasks ranges in size from 10-25K pairs (Table 5.1 and Table 5.2). The coverage of languages is limited by the availability of parallel names. Also the transliteration generation systems rely on parallel names in order to develop and train the systems. The goal of the name transliteration mining task is to extract name transliterations from parallel texts. In the NEWS 2010 transliteration mining shared task [Kumaran et al., 2010], a mining system is trained on a small list of transliterations for a pair of source and target languages. The list is called "seed data", typically contains about 1K name pairs. The trained system then identifies single word transliterations in standard standard interlinked Wikipedia topics, called "WIL or Wikipedia Interlanguage Links"[1]. The links connect articles on the same topic in multiple languages.

The task includes five language pairs: English-Chinese (WM-EnCh), English-Hindi (WM-EnHi), English-Tamil (WM-EnTa), English-Russian (WM-EnRu), and English-Arabic

[1] `http://en.wikipedia.org/wiki/Help:Interlanguage_links`

(WM-EnAr). Each WIL data set consists of pairs of topic titles in the source and target languages. The task is to identify parts of the topics that are indeed transliterations. It is possible that a WIL's title may contain 0, 1 or more transliteration pairs. The number of title pairs ranges between 10K and 200K titles. The task data do not include examples of transliterations extracted from the WIL titles. The models must be learned purely from the seed sets. The seed sets are not necessarily from the Wikipedia domain. The test sets, however, are subsets of WIL data which have been hand labeled for evaluation. The test sets comprise around 1K examples of WIL title pairs.

Precision (P), recall (R), and F-score ($F_1$) are the evaluation matrices for the transliteration mining task. These evaluation matrices are calculated from the numbers of true/false-positive/negatives as follows:

$$P = \frac{TP}{TP + FP} \tag{5.10}$$

$$R = \frac{TP}{TP + FN} \tag{5.11}$$

$$F_1 = \frac{2 * P * R}{P + R} \tag{5.12}$$

where,

- the true positives, $TP$, are the number of pairs that are identified by the system as transliterations and which are also tagged in the gold standard transliteration pairs.
- the false positives, $FP$, are the number of pairs that are identified by the system as transliterations but which are not tagged in the gold standard transliteration pairs.
- the false negative, $FN$, are the number of pairs that are not identified by the system as transliterations but which are tagged in the gold standard as transliteration pairs.

### 5.2.1 Approaches to transliteration mining

For transliteration mining, approaches can be broadly categorized into (1) classification-based and (2) generation-based methods. Classification-based approaches [Bergsma and Kondrak, 2007, Klementiev and Roth, 2006] aim to build a binary classifier to identify whether a candidate pair consisting of a source word $s$ and a target word $t$ is a transliteration pair. Training binary classifiers requires positive and negative examples, i.e., pairs that are transliterations and pairs that are not transliterations. For the NEWS 2010 task, the seed data provide transliteration examples, and these examples can be used as positive training examples. For negative training examples, one possible approach is to randomly select unaligned target names in the seed data for each source name [Klementiev and Roth, 2006].

This random sampling method potentially generates "easy" negative examples which are clearly not transliterations. To create "competitive" negative examples, Bergsma and Kondrak [2007] proposed selecting negatives based on a pair's Longest Common Subsequence Ratio (LCSR). Only negative examples that have an LCSR above a threshold are included as training examples.

Bergsma and Kondrak [2007] train a binary SVM classifier using substring alignment features. The reported results are promising; however, the features are limited to only those substrings that are below a certain maximum size (three or four characters). This is because the number of unique features increases exponentially as the substring size increases. To extend this approach, one can use a string kernel function instead of an explicit feature representation. One of the systems that the University of Alberta submitted to the NEWS 2010 transliteration mining shared task [Jiampojamarn et al., 2010b] is based on an SVM model using a standard $n$-gram string kernel. The kernel function is defined as the total number of common $n$-grams that appear in both strings $s$ and $t$. Each $n$-gram count is weighted by its length — a factor of $\lambda^n$. The maximum length of $n$-grams and the weight factor $\lambda$ are optimized using cross-validation.

A simple approach is to compute the string similarity between two strings via Normalized Edit Distance (NED). NED is the edit distance function divided by the maximum length of the two strings. NED is thus always between 0 and 1. Instead of expressing NED as a distance function, a similarity expression can be achieved by subtracting the original fraction from 1. In this way, NED is equal to 1 if the two strings are exactly identical; and 0 otherwise. Since the source and target languages may use different writing scripts, Romanization is required for non-Latin scripts. A simple Romanization table can be obtained by extracting the highest conditional probabilities from character alignments in seed data. The character alignments can be generated with the m2m-aligner [Jiampojamarn et al., 2007] but limiting the alignment size to be exactly one on both the source and target side. Essentially, this alignment model is similar to the model of Ristad and Yianilos [1998]. Although the simple Romanization table is not highly accurate, it requires no language-specific knowledge. This approach can be viewed as a way to transliterate the target language name to the source language name via a Romanization table. Then, the system's decisions are based on NED similarity. A transliteration is proposed if the NED value is above some threshold. Without development sets, these thresholds are chosen according to the average word length in the seed sets. The thresholds are higher for longer average word lengths and vice versa.

Generation-based approaches, on the other hand, generate transliterations for source or

target names and compares the generated names to the target or source names in the candidate list. Darwish [2010] proposed a generation-based transliteration mining system. The system uses a generative Hidden Markov Model (HMM) to generate target language names given source language names. The system identifies a candidate pair ($s$,$t$) as a transliteration if the generation of $t$ given $s$ exactly matches the string $s$. This exact match criteria provides high precision but low recall. To improve recall, a modified SOUNDEX scheme is used when vowel mismatch is discarded and similar characters are conflated. The modified SOUNDEX method thereby relaxes the constraint of exact match when the system makes decisions. Recall is also further improved by using the transliteration pairs, found by the original model, to re-train the HMM transliteration model.

Noeman and Madkour [2010] proposed a generation-based approach inspired by phrase-based statistic machine translation [Koehn et al., 2003]. The system learns alignment model from the seed data using GIZA++ [Och and Ney, 2003]. The list of character alignments is represented as a finite state automaton. For a given source name, candidate transliterations are generated within a certain $k$ edits from the model output, and these candidates are compared to the target language words.

### 5.2.2 Application of DirecTL+ to transliteration mining

A generation based approach for transliteration mining is to determine if the generated transliterations of a source word $\hat{t} = F(s)$ and a target word $\hat{s} = B(t)$ are similar to their corresponding words in a candidate pair $(s, t)$. The state-of-the-art transliteration system DirecTL+ can be applied to generate both forward and backward transliterations. The system is trained on the seed sets which contain small lists, approximately 1K of parallel names. This training creates the $F(s)$ and $B(t)$ models. To decide if the given candidate pair $(s, t)$ is a transliteration pair, a score function (Equation 5.13) is calculated. The candidate pair $(s, t)$ is proposed as a transliteration if $Sim(s, t) > \tau$.

$$Sim(s,t) = \frac{w_1 \cdot NED(t,\hat{t}) + w_2 \cdot NED(s,\hat{s})}{w_1 + w_2} \tag{5.13}$$

$NED(t, \hat{t})$ is the normalized edit distance between strings $t$ and $\hat{t}$. The $NED$ values are expressed in a score function rather than a cost function by subtracting the edit distance values from 1. $w_1$ and $w_2$ are combination weights that favor the forward and backward transliteration models. Ideally, $\tau, w_1$ , and $w_2$ should be optimized on a development set. Since the task provides no development set, one can optimize these parameters by creating negative examples [Klementiev and Roth, 2006] and taking positive examples from the seed

sets. However, this optimization is highly reliant on the seed sets whose data may not have the same characteristics as the WIL's titles. Alternatively, one can model the NED of word pairs as a mixture model [Dinov, 2008] or consider the problem as clustering with two clusters.

One important component in the generation-based approach is the extraction of source-target candidate pairs from the WIL's titles. Ultimately, the overall performance on the end task depends on the quality of the candidate pairs extracted in the first place. For languages with explicit tokenization (e.g. English and Russian), word segmentation can be performed using sequences of one or more spaces, as well as punctuation symbols including hyphens, underscores, brackets, and other non-alphanumeric characters. The candidate pairs are constructed by taking the cross product of the source and target language words within the aligned WIL titles. However, for a language like Chinese when transliterated names consist of multiple Chinese characters and when these names are not explicitly separated from other text, word segmentation becomes problematic. Fortunately, some WIL's titles include a separation symbol "·" on the target, Chinese side. In this case, word segmentation can be based on the separation symbol. On other hand, when there is no indication of word segmentation, Chinese candidate names can be constructed from all possible $n$-grams ($2 \leq n \leq L$), where $L$ is the length of the Chinese title under consideration. The highest similarity score between these $n$-grams and the source words, where the score is also above a threshold value, is taken to indicate a transliteration pair. Like in the transliteration generation tasks, a Pinyin representation can be used as a source of additional information. Given each English name, separate transliteration models generate Chinese characters and Pinyin. The Chinese target names in the candidate list are also converted to Pinyin for comparison with the generated outputs. For computing the similarity for non-pinyin output, Chinese characters with similar sounds must be considered mismatches. The Pinyin representation helps to identify similar sounds in the generated and candidate words.

### 5.2.3 Summary of evaluation and results

Table 5.6 reports the system performance evaluated on the NEWS 2010 transliteration mining shared tasks [Kumaran et al., 2010]. These results were submitted to the shared tasks as University of Alberta submissions. NED refers to the simple system that is based on the Normalized Edit Distance between the romanization of the source and target candidate words. NED also incorporates sets of simple rules to identify non-transliteration pairs. The rules include word lengths, capitalization, and numerical usage. Since there is no de-

velopment set provided, the threshold values are set based on the average word length in the seed sets. The values are 0.38, 0.48, 0.52 and 0.58 for WM-EnHi, WM-EnAr, WM-EnTa, and WM-EnRu. Bergsma and Kondrak [2007] is the SVM classifier with a linear kernel function. The system uses features derived from alignments generated by the m2m-aligner [Jiampojamarn et al., 2007]. StringKernel is an improved system that trains with the string kernel function. DirecTL+ is a basic system that uses a fixed threshold value ($\tau$) of 0.58 and equally weights the forward and backward models in the linear combination ($w_1 = w_2 = 1$) for all data sets. "+ average cutting" indicates when the threshold value $\tau$ is set at the point of average normalized edit distance plus a standard deviation ($\tau = Avg. + SD$). "+ oracle cutting" indicates that the threshold values is set at the peak point of F-score performance. The "Oracle candidate" results are the upper bound of the provided candidate list assuming the perfect precision performance. "StringMatch" indicates the WM-EnCh system that utilizes the $n$-gram matching method instead of the simple tokenization used in the other language sets.

The NED system is simple but it achieves good performance across different language pairs. It achieves the best result for the WM-EnRu task at 87.5% F-score. For all data sets, StringKernel shows significant improvements over the system of Bergsma and Kondrak [2007]. These improvements indicate the power of using the string kernel function instead of a linear representation for these tasks. The DirecTL+ performance depends on the quality of the threshold value used to discriminate true transliteration pairs from non-transliterations. A threshold value of 0.58 is based on work mining cognates from word-aligned bitexts, see [Bergsma and Kondrak, 2007, Melamed, 1999] This threshold provides the best results on the WM-EnRu task compared to performance at the "oracle cutting" points. The average cutting method is a simple but efficient way to find the optimal threshold for most language pairs. The difference in F-score between average cutting and oracle cutting is less than 1% F-score for all language pair except WM-EnRu.

Figure 5.1 shows the histogram of NED values on the WM-EnAr candidate set. The x-axis is the NED value and the y-axis is the number of occurrences in the set with 0.05 precision at each step. The 3rd degree polynomial approximation function suggests that there are two distributions mixed in the data. The first vertical line from the left indicates the cutting point at the average NED value. Each vertical line after the first is the cutting point at the average NED value with an additional 0.5 standard deviations. Changing the threshold value from left to right affects the trade-off between precision and recall. In general, using the far left point as the threshold results in high recall but low precision. The

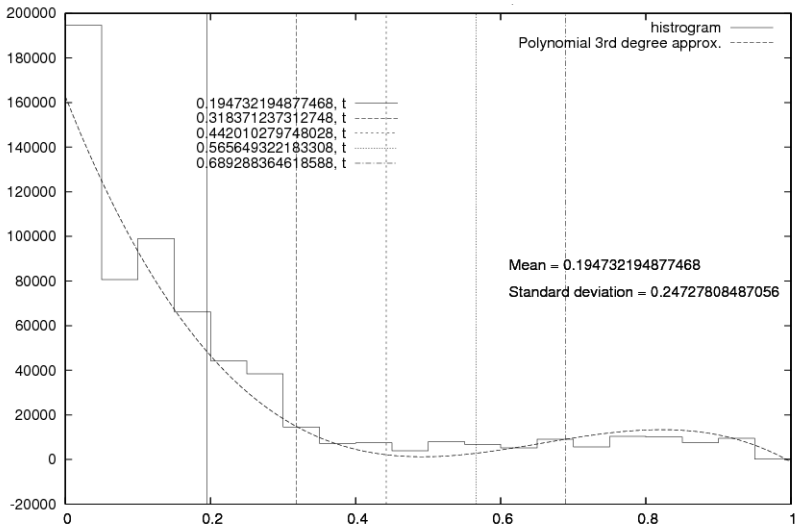


Figure 5.1: WM-EnAr histogram.

overall F-score results for each vertical line from left to right are 72.0%, 82.3%, 84.3%, 82.8% and 75.2% respectively.

Table 5.7 reports the results of using DirecTL+ with the average cutting method and also alternative clustering methods on the WM-Ar task. The average cutting method simply sets the cut-off threshold to the average of all similarity scores plus their standard deviation. The alternative methods consider the separation task as a clustering problem. "1-d Beta mixture model" divides the data into 2 clusters (transliterations and non-transliterations) such that the clusters are best fit to two beta distributions. Similarly, "1-d Gaussian mixture model" uses the Gaussian distribution as the base function. "Multivariate Gaussian mixture model" uses the original NED scores of the forward and backward models as mixture dimensions instead of the combination score as given in Equation 5.13. "Simple K-means" is a widely used algorithm for clustering problems. It creates two clusters such that the summed Euclidean distances between each data point and its cluster centroid is minimized. The simple average cutting method shows better overall performance than the other techniques. The linear combination score yields better performance than modeling each score in the multivariate mixture model. These results demonstrate the effectiveness of the average cutting point for separating true transliterations from non-transliterations. Although the average cutting method achieves results that are close to the optimal results with respect to the similarity function and threshold values, more sophisticated techniques and different approaches that are not based on these threshold values are required to further improve the clustering process.

The simple method for word segmentation is sufficient for creating candidate lists in most languages, but not for WM-EnCh. The "oracle candidate" results for WM-EnRu, WM-EnHi, and WM-EnTa are all above 99% F-score while the method achieves 95.5% F-score on WM-EnAr. In general, for languages with words that can be tokenized by non-alphabetic symbols, there is no difficulty in obtaining candidate word lists from the WIL titles. On the other hand, the simple word tokenization method achieves only 22% oracle F-score on the Chinese data. This low oracle performance prevents the baseline DirecTL+ system from obtaining a reasonable result, as the candidate pairs are so low-quality in the first place. The generation model of the StringMatch method is based on the DirecTL+ system; however, it searches for the most likely transliteration pairs from the WIL titles by greedily matching each source word to every possible $n$-gram of Chinese target Pinyin, from left to right. This method achieves clearly superior accuracy. StringMatch achieved 53% F-score which is substantially lower than other results obtained in other languages. These performance gaps illustrate the complexity of a non-alphabetic language, and suggest that the DirecTL+ generation and string similarity function are more suitable for alphabetic languages than logosyllabic languages. Further research on non-alphabetic languages is necessary to improve the overall performance.

Comparing to other reported results in the shared task [Kumaran et al., 2010], there is no single approach that achieves the best results for all test sets. The NED system achieves the best WM-EnRu result while StringKernel obtains the best WM-EnTa result. The DirecTL+ system with the average cutting method outperforms both NED and StringKernel systems on the WM-EnHi and WM-EnAr tasks; however, the best results reported on these sets are from MINT for WM-EnHi [Udupa et al., 2008] (which trains on additional seed data and achieves a 94.4% F-score) and from the system of Noeman and Madkour [2010] for WM-EnAr (91.5% F-score).

## 5.3 Summary

I presented the DirecTL model for name transliteration, including applications to both transliteration generation and mining. The transliteration generation task has the same basic principles as grapheme-to-phoneme conversion. The baseline idea of using DirecTL is to replace the phoneme sequences in G2P with the target language symbols in transliteration. A method for multiple outputs is proposed to exploit examples where one source language name may correspond to multiple correct target language names. The NEWS gen-

eration shared task results suggest that DirecTL is highly language-independent; it achieves state-of-the-art performance without requiring additional resources nor language-specific knowledge. However, making use of Chinese Pinyin and Jaso Korean are language-specific modifications that improve performance over the baseline DirecTL systems. In both the NEWS 2009 and NEWS 2010 transliteration shared tasks, DirecTL systems obtained either the first or second best performance for most evaluated data sets. These successful results confirm the power and generality of many-to-many alignments and the online discriminative training framework.

The DirecTL+ system for transliteration mining is a generation-based approach. The key idea is to generate transliterations for both source and target language words. The identification of transliteration pairs is accomplished by seeing how similar the generated words are to the candidates. The proposed approach achieves either comparable or better results than a range of other systems including the NED, StringKernel and Bergsma and Kondrak [2007] systems. The word segmentation problem in Chinese prevents simple tokenization-based approaches from achieving acceptable quality in the extracted candidate word lists. However, the greedy string-matching method obtains promising results, showing the effectiveness of a DirecTL system when a better quality set of extracted candidate words is available.

| Task | System | F-score | Precision | Recall |
|---|---|---|---|---|
| WM-EnRu | NED | 87.5 | 88.0 | 86.9 |
| | Bergsma and Kondrak [2007] | 77.8 | 68.4 | 90.2 |
| | StringKernel | 81.1 | 74.6 | 88.9 |
| | DirecTL+ | 79.3 | 77.3 | 81.5 |
| | + average cutting | 78.0 | 68.3 | 91.0 |
| | + oracle cutting | 80.2 | 72.4 | 90.0 |
| | Oracle candidate | 99.4 | 100 | 98.8 |
| WM-EnHi | NED | 90.7 | 87.5 | 94.1 |
| | Bergsma and Kondrak [2007] | 88.2 | 88.3 | 88.0 |
| | StringKernel | 92.4 | 95.4 | 89.5 |
| | DirecTL+ | 91.4 | 96.6 | 86.7 |
| | + average cutting | 93.6 | 94.0 | 93.1 |
| | + oracle cutting | 93.7 | 90.6 | 96.9 |
| | Oracle candidate | 99.7 | 100 | 99.4 |
| WM-EnTa | NED | 79.1 | 91.6 | 69.6 |
| | Bergsma and Kondrak [2007] | 82.9 | 80.8 | 85.2 |
| | StringKernel | 91.4 | 92.3 | 90.6 |
| | DirecTL+ | 86.0 | 95.4 | 78.3 |
| | + average cutting | 91.2 | 89.7 | 92.6 |
| | + oracle cutting | 91.9 | 89.3 | 94.8 |
| | Oracle candidate | 99.8 | 100 | 99.7 |
| WM-EnAr | NED | 80.0 | 81.8 | 78.3 |
| | Bergsma and Kondrak [2007] | 81.6 | 83.4 | 79.8 |
| | StringKernel | 82.7 | 91.7 | 75.3 |
| | DirecTL+ | 82.6 | 88.1 | 77.8 |
| | + average cutting | 84.3 | 81.8 | 87.0 |
| | + oracle cutting | 84.7 | 83.1 | 86.3 |
| | Oracle candidate | 95.5 | 100 | 91.4 |
| WM-EnCh | StringMatch | 53.0 | 69.8 | 42.7 |
| | DirecTL+ | 0.09 | 0.45 | 0.05 |
| | Oracle candidate | 22.0 | 100 | 12.2 |

Table 5.6: Transliteration mining results.

| System | F-score | Precision | Recall |
|---|---|---|---|
| Average cutting | 84.3 | 81.8 | 87.0 |
| 1-d Beta mixture model | 80.0 | 72.8 | 88.8 |
| 1-d Gaussian mixture model | 78.2 | 69.8 | 88.9 |
| Multivariate Gaussian mixture model | 76.8 | 66.8 | 90.5 |
| Simple K-means | 83.2 | 79.4 | 87.4 |

Table 5.7: DirecTL+ with average cutting and other clustering methods on the WM-EnAr task

# Chapter 6

# Conclusion

In this thesis, I presented approaches to the grapheme-to-phoneme conversion and name transliteration tasks. My contributions can be directly applied in two main applications: (1) text-to-speech and (2) machine translation. In text-to-speech application, pronunciation of common words can be largely found in a lexicon inventory. Automatic phoneme generation is required only when words to be synthesized are not listed in the inventory. The DirecTL system plays an important role in synthesizing these words including proper names, misspellings and uncommon words. In machine translation application, the DirecTL system can be applied to transliterate proper names that are left from a translation model due to the fact that these proper names cannot be translated by their meanings.

My research contributions are in both alignment and generation techniques. I first proposed the many-to-many alignment algorithm to improve over the one-to-one alignments that have been widely used in G2P conversion. The many-to-many alignment algorithm relaxes the one-to-one constraint that limits the size of grapheme and phoneme sequences being aligned. I investigated several alternative alignment methods in attempting to improve both alignment and generation performances. These alternative methods include phonetic alignment, ALINE, based on [Kondrak, 2000], integer linear programming–inspired by the minimal model of [Ravi and Knight, 2009], and alignment-by-aggregation approaches. I conducted an in-depth study to evaluate these alignment techniques compared to the existing one-to-one alignment methods.

To incorporate the many-to-many alignments, I presented grapheme chunking and post-processing language models that incorporate the many-to-many alignments into existing classifiers. The proposed approach connects each component in a pipeline framework. I demonstrated that the proposed system outperforms existing systems based on one-to-one alignments, although the proposed pipeline framework potentially allows a propagation

of errors. I collapsed the pipeline framework and proposed joint processing and online discriminative training for G2P. The joint approach simultaneously finds the most likely grapheme subsequences that generate the most likely phoneme sequence outputs via a phrase-based decoder [Zens and Ney, 2004]. I presented the online discriminative training framework that is based on the online large margin training technique [Crammer and Singer, 2003]. The online large margin technique significantly improves performance over the simple averaged perceptron of Collins [2002]. I combined the proposed G2P approach with a state-of-the-art stress assignment system and demonstrated several combination techniques, including the joint, pre-processing and post-processing methods. I further investigated a training paradigm that requires no alignments in the training data; this training method is inspired by the end-to-end approach of Liang et al. [2006]. The experimental results demonstrated that the end-to-end approach is no better than the joint approach that incorporates the many-to-many alignments. I also integrated the generative joint $n$-gram approach [Bisani and Ney, 2008] into the discriminative approach via a feature template. Additionally, I proposed an approximate beam search in place of the exact phrase-based decoder and showed that the integrated system outperforms both the generative joint $n$-gram and the original discriminative approaches.

I applied this successful G2P system to name transliteration tasks in the NEWS 2009 and 2010 Machine Transliteration Shared Tasks [Kumaran et al., 2010, Li et al., 2009, 2010]. I developed language-independent transliteration generation systems. I evaluated the language-independent systems as well as ones with language-specific pre-processing. The experimental results illustrated that the pre-processing does not always help in several language pairs. The language-specific processes only help to better represent training data for alignment and generation tasks for certain writing systems; e.g., Chinese, Korean and Japanese. I proposed a transliteration mining system that is based on the generation system in which similarity is measured between generated transliterations and transliteration candidate words. The final system demonstrated state-of-the-art performance in the NEWS shared tasks in both transliteration generation and transliteration mining.

**Future Work**

In Chapter 3, I proposed a many-to-many alignment method to improve on the existing one-to-one alignment methods and later proposed alternative alignment methods to further improve the EM-based alignment algorithm. One research direction is to investigate the use of Bayesian inference [Chiang et al., 2010, Goldwater and Griffiths, 2007] for grapheme-

to-phoneme alignments as well as for name transliteration alignments. An ideal alignment model should provide a compact model size, and good data likelihood. While the EM-based alignment approaches optimize directly to obtain the highest data likelihood, it suffers from creating unnecessary alignment links and overfits the training data. Another research direction is to investigate a discriminative training method for finding alignments and also to extend it to a higher Markov model [McCallum et al., 2005].

The DirecTL systems proposed in Chapter 4 are based on character substring features in both input and output spaces. The learning model obtains information from substring evidence in the training data without linguistic information. For G2P, stress, syllabification and morphological constraints can help to further improve the overall performance [Bartlett et al., 2008, Demberg et al., 2007, Marchand and Damper, 2006]. It is an interesting research direction to further investigate useful linguistic features and incorporate them in the generation model. One potential approach is to re-rank the current system output using full-word linguistic information. The experimental results shown in Figure 4.7 indicate that 98% of correct sequence outputs can be achieved in the top 50 answers. from a model trained on only half of the training data. One difficulty in this future work is how to re-rank the $n$-best list output in a way that more correct answers appear at the top of the list. Among others, linguistic information presents a promising set of features that can guide the re-ranking model to obtain better performance. In addition to linguistic information, another possible way to re-rank the $n$-best outputs is to observe the similarity of each output sequence with other sequences in the $n$-best list, i.e., to apply Minimum Bayes-risk decoding [Goel and Byrne, 2000]. This approach has helped automatic speech recognition [Goel and Byrne, 2000] and machine translation [Kumar and Byrne, 2004]. However, it showed no improvement in dependency parser [Smith and Smith, 2007]. It is worth to investigate if Minimum Bayes-risk decoding can help grapheme-to-phoneme conversion and name transliteration tasks.

In Chapter 5, I presented a generation-based approach for the name transliteration mining tasks. The system's performance is largely dependent on two factors: (1) the quality of extracted candidate lists, and (2) choosing an optimal threshold value. With regards to the first factor, the simple tokenization-based approach provides reasonable candidate lists for several languages where the word segmentation problem does not exist in their writing systems. However, simple methods provide ineffective results in Chinese. Word segmentation [Jiang et al., 2008, Zhang and Clark, 2008] for these languages will be required in order to obtain good quality in the extracted candidate lists. In addition, the best

performance achieved in WM-EnCh with the StringMatch method is significantly lower than the other languages leaving opportunities in future works to improve not only the segmentation but also the generation and string similarity functions. Regarding choosing an optimal threshold, while the simple average cutting method (Section 5.2.3) provides near-optimal solutions, it would be interesting to find the global optimal point that separates non-transliteration pairs and true transliteration pairs without requiring training or development sets. In this thesis, I illustrated how to extend the DirecTL framework for transliteration mining. The proposed technique is simple but yet effective to achieve the state-of-the-art performance. In future works, it is interesting to explore other sophisticated techniques that directly aim for transliteration mining.

# Bibliography

David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991. ISSN 0885-6125.

Eiji Aramaki and Takeshi Abekawa. Fast decoding and easy implementation: Transliteration as sequential labeling. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 65–68. Association for Computational Linguistics, 2009.

Harald Baayen, Richard Piepenbrock, and Leon Gulikers. The CELEX2 lexical database. LDC96L14, 1996.

Paul C. Bagshaw. Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression. *Computer Speech and Language*, 12(2): 119–142, 1998.

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 568–576. Association for Computational Linguistics, 2008.

Shane Bergsma and Grzegorz Kondrak. Alignment-based discriminative string similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 656–663. Association for Computational Linguistics, 2007.

Shane Bergsma and Qin Iris Wang. Learning noun phrase query segmentation. In *EMNLP-CoNLL*, pages 819–826. Association for Computational Linguistics, 2007.

Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2 edition, 1999.

Aditya Bhargava and Grzegorz Kondrak. Language identification of names with SVMs. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 693–696. Association for Computational Linguistics, 2010.

Maximilian Bisani and Hermann Ney. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 105–108, 2002.

Maximilian Bisani and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451, 2008. ISSN 0167-6393.

Alan W. Black, Kevin Lenzo, and Vincent Pagel. Issues in building general letter to sound rules. In *The Third ESCA Workshop in Speech Synthesis*, pages 77–80, 1998.

Antal Van Den Bosch and Sander Canisius. Improved morpho-phonological sequence processing with constraint satisfaction inference. *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology, SIGPHON '06*, pages 41–49, 2006.

Antal Van Den Bosch and Walter Daelemans. Do not forget: Full memory in memory-based learning of word pronunciation. In *Proceedings of NeMLaP3/CoNLL98*, pages 195–204, Sydney, Australia, 1998.

Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Math. Program.*, 63(2):129–156, 1994. ISSN 0025-5610.

Yair Censor and Stavros A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1997. ISBN 978-0195100624.

Stanley F. Chen. Conditional and joint models for grapheme-to-phoneme conversion. In *Proceedings of the Eurospeech 2003*, 2003.

David Chiang, Jonathan Graehl, Kevin Knight, Adam Pauls, and Sujith Ravi. Bayesian inference for finite-state transducers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 447–455. Association for Computational Linguistics, 2010.

N. Chomsky and M Halle. The sound pattern of English. *Harper and Row, New York*, 1968.

John Coleman. Improved prediction of stress in out-of-vocabulary words. In *IEEE Seminar on the State of the Art in Speech Synthesis*, 2000.

Michael Collins. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, 2002.

A. Content, P. Mousty, and M. Radeau. Brulex. une base de données lexicales informatisée pour le franais écrit et parlé (Brulex, a lexical database for written and spoken French). *L' Année Psychologique*, 90:551–566, 1990.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991, 2003. ISSN 1533-7928.

Walter Daelemans and Antal Van Den Bosch. Language-independent data-oriented grapheme-to-phoneme conversion. In *Progress in Speech Synthesis*, pages 77–89. Springer-Verlag, New York, USA, 1997.

Walter Daelemans, Jakub Zavrel, Ko Van Der Sloot, and Antal Van Den Bosch. TiMBL: Tilburg Memory Based Learner, version 5.1, reference guide. In *ILK Technical Report Series 04-02*, 2004.

R. Damper and J. Eastmond. Pronunciation by analogy: impact of implementational choices on performance. *Language and Speech, 40(1):1–23*, 1997.

R. I. Damper, Y. Marchand, M. J. Adamson, and K. Gustafson. Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches. *Computer Speech and Language*, 13(2):155–176, 1999.

Robert I. Damper, Yannick Marchand, John DS. Marsters, and Alexander I. Bazin. Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology*, 8(2):147–160, 2005.

Kareem Darwish. Transliteration mining with phonetic conflation and iterative training. In *Proceedings of the 2010 Named Entities Workshop*, pages 53–56. Association for Computational Linguistics, 2010.

Amitava Das, Asif Ekbal, Tapabrata Mondal, and Sivaji Bandyopadhyay. English to Hindi machine transliteration system at NEWS 2009. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 80–83. Association for Computational Linguistics, 2009.

Amitava Das, Tanik Saikh, Tapabrata Mondal, Asif Ekbal, and Sivaji Bandyopadhyay. English to Indian languages machine transliteration system at NEWS 2010. In *Proceedings of the 2010 Named Entities Workshop*, pages 71–75. Association for Computational Linguistics, 2010.

Vera Demberg, Helmut Schmid, and Gregor Möhler. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 96–103, 2007.

Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the EM algorithm. In *Journal of the Royal Statistical Society*, pages B:1–38, 1977.

Ivo Dinov. Expectation maximization and mixture modeling tutorial. In *UC Los Angeles: Statistics Online Computational Resource*, 2008. URL `http://escholarship.org/uc/item/1rb70972`.

Qing Dou. An SVM ranking approach to stress assignment. Master's thesis, University of Alberta, 2009.

Qing Dou, Shane Bergsma, Sittichai Jiampojamarn, and Grzegorz Kondrak. A ranking approach to stress prediction for letter-to-phoneme conversion. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 118–126. Association for Computational Linguistics, 2009.

Andre Elisseeff and Jason Weston. Kernel methods for multi-labelled classification and categorical regression problems. In *In Advances in Neural Information Processing Systems 14*, pages 681–687, 2001.

H. Elovitz, R. Johnson, A. McHugh, and J. Shore. Letter-to-sound rules for automatic translation of English text to phonetics. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(6):446–459, 1976.

Herman Engelbrecht and Tanja Schultz. Rapid development of an Afrikaans-English speech-to-speech translator. In *International Workshop of Spoken Language Translation (IWSLT)*, Pittsburgh, PA, USA, 2005.

Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

Andrew Finch and Eiichiro Sumita. Transliteration by bidirectional statistical machine translation. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 52–56. Association for Computational Linguistics, 2009.

Andrew Finch and Eiichiro Sumita. Transliteration using a phrase-based statistical machine translation system to re-score the output of a joint multigram model. In *Proceedings of the 2010 Named Entities Workshop*, pages 48–52. Association for Computational Linguistics, 2010.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296, 1999.

Nagendra Goel, Samuel Thomas, Mohit Agarwal, Pinar Akyazi, Lukas Burget, Kai Feng, Arnab Ghoshal, Ondrej Glembek, Martin Karafiat, Daniel Povey, Ariya Rastrow, Richard C. Rose, and Petr Schwarz. Approaches to automatic lexicon learning with limited training examples. In *The 35th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

Vaibhava Goel and William J. Byrne. Minimum bayes risk automatic speech recognition. In *Computer Speech and Language*, pages 115–135, 2000.

Sharon Goldwater and Tom Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751. Association for Computational Linguistics, 2007.

Gumwon Hong, Min-Jeong Kim, Do-Gil Lee, and Hae-Chang Rim. A hybrid approach to English-Korean name transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 108–111, Suntec, Singapore, August 2009. Association for Computational Linguistics.

International Phonetic Association. *Handbook of the International Phonetic Association*. Cambridge University Press, 1999.

Martin Jansche and Richard Sproat. Named entity transcription with pair n-gram models. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 32–35, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Sittichai Jiampojamarn and Grzegorz Kondrak. Letter-phoneme alignment: An exploration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 780–788, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, USA, 2007.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. Joint processing and discriminative training for letter-to-phoneme conversion. In *The 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 905–913, Columbus, OH, USA, 2008.

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. DirecTL: a language independent approach to transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 28–31. Association for Computational Linguistics, 2009.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. Integrating joint n-gram features into a discriminative training framework. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 697–700. Association for Computational Linguistics, 2010a.

Sittichai Jiampojamarn, Kenneth Dwyer, Shane Bergsma, Aditya Bhargava, Qing Dou, Mi-Young Kim, and Grzegorz Kondrak. Transliteration generation and mining with limited training resources. In *Proceedings of the 2010 Named Entities Workshop*, pages 39–47. Association for Computational Linguistics, 2010b.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904. Association for Computational Linguistics, 2008.

Thorsten Joachims. Making large-scale support vector machine learning practical. In *Advances in kernel methods: support vector learning*, pages 169–184. MIT Press, 1999.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

Anne K. Kienappel and Reinhard Kneser. Designing very compact decision trees for grapheme-to-phoneme transcription. In *EUROSPEECH-2001*, pages 1911–1914, 2001.

Alexandre Klementiev and Dan Roth. Named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 82–88. Association for Computational Linguistics, 2006.

Kevin Knight and Jonathan Graehl. Machine transliteration. *Computational Linguistics*, 24 (4):599–612, 1998.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54. Association for Computational Linguistics, 2003.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180. Association for Computational Linguistics, 2007.

John Kominek and Alan W Black. Learning pronunciation dictionaries: Language complexity and word selection strategies. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 232–239, 2006.

Grzegorz Kondrak. A new algorithm for the alignment of phonetic sequence. In *The first Meeting of the North American Chapter of the Association for Computational Liguistics*, pages 288–295. Association for Computational Linguistics, 2000.

Shankar Kumar and William Byrne. Minimum bayes-risk decoding for statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 169–176. Association for Computational Linguistics, 2004.

A Kumaran, Mitesh M. Khapra, and Haizhou Li. Report of NEWS 2010 transliteration mining shared task. In *Proceedings of the 2010 Named Entities Workshop*, pages 21–28, Uppsala, Sweden, 2010. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., 2001.

Haizhou Li, Min Zhang, and Jian Su. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 159–166, 2004.

Haizhou Li, A Kumaran, Vladimir Pervouchine, and Min Zhang. Report of NEWS 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 1–18. Association for Computational Linguistics, 2009.

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. Report of NEWS 2010 transliteration generation shared task. In *Proceedings of the 2010 Named Entities Workshop*, pages 1–11. Association for Computational Linguistics, 2010.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768. Association for Computational Linguistics, 2006.

Yannick Marchand and Robert I. Damper. A multistrategy approach to improving pronunciation by analogy. *Computational Linguistics*, 26(2):195–219, 2000.

Yannick Marchand and Robert I. Damper. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*, 13(1):1–24, 2006. ISSN 1351-3249.

Andrew McCallum, Kedar Bellare, and Fernando C. N. Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. In *UAI*, pages 388–395, 2005.

I. Dan Melamed. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130, 1999. ISSN 0891-2017.

Sara Noeman. Language independent transliteration system using phrase-based SMT approach on substrings. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 112–115. Association for Computational Linguistics, 2009.

Sara Noeman and Amgad Madkour. Language independent transliteration mining system using finite state automata framework. In *Proceedings of the 2010 Named Entities Workshop*, pages 57–61. Association for Computational Linguistics, 2010.

Franz Josef Och. Minimum error rate training in statistical machine translation. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Jong-Hoon Oh, Kiyotaka Uchimoto, and Kentaro Torisawa. Machine transliteration using target-language grapheme and phoneme: Multi-engine transliteration approach. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 36–39. Association for Computational Linguistics, 2009.

Steve Pearson, Roland Kuhn, Steven Fincke, and Nick Kibre. Automatic methods for lexical stress assignment and syllabification. In *ICSLP*, pages 423–426, 2000.

Vladimir Pervouchine, Haizhou Li, and Bo Lin. Transliteration alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 136–144. Association for Computational Linguistics, 2009.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

Taraka Rama and Karthik Gali. Modeling machine transliteration as a phrase based statistical machine translation problem. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 124–127. Association for Computational Linguistics, 2009.

Taraka Rama, Anil Kumar Singh, and Sudheer Kolachina. Modeling letter-to-phoneme conversion as a phrase based statistical machine translation problem with Minimum Error Rate training. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 90–95. Association for Computational Linguistics, 2009.

Sujith Ravi and Kevin Knight. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 504–512. Association for Computational Linguistics, 2009.

Korin Richmond, Robert A. J. Clark, and Sue Fitt. Robust LTS rules with the Combilex speech technology lexicon. In *Proceedings of Interspeech*, pages 1295–1298, 2009.

Eric Sven Ristad and Peter N. Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

Juergen Schroeter, Alistair Conkie, Ann Syrdal, Mark Beutnagel, Matthias Jilka, Volker Strom, Yeon-Jun Kim, Hong-Goo Kang, and David Kapilow. A perspective on the next challenges for TTS research. In *IEEE 2002 Workshop on Speech Synthesis*, 2002.

Terrence J. Sejnowski and Charles R. Rosenberg. Parallel networks that learn to pronounce English text. In *Complex Systems*, pages 1:145–168, 1987.

Tarek Helmy Sherif. Substring-based transliteration. Master's thesis, Department of Computing Science, University of Alberta, 2007.

Praneeth Shishtla, Surya Ganesh V, Sethuramalingam Subramaniam, and Vasudeva Varma. A language-independent transliteration schema using character aligned models at NEWS 2009. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 40–43. Association for Computational Linguistics, 2009.

David A. Smith and Noah A. Smith. Probabilistic models of nonprojective dependency trees. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 132–140. Association for Computational Linguistics, 2007.

Yan Song, Chunyu Kit, and Xiao Chen. Transliteration of name entity via improved statistical translation on character sequences. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 57–60. Association for Computational Linguistics, 2009.

Yan Song, Chunyu Kit, and Hai Zhao. Reranking with multiple features for better transliteration. In *Proceedings of the 2010 Named Entities Workshop*, pages 62–65, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Richard Sproat, Tao Tao, and ChengXiang Zhai. Named entity transliteration with comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 73–80. Association for Computational Linguistics, 2006.

J. Suontausta and J. Tian. Low memory decision tree method for text-to-phoneme mapping. *Automatic Speech Recognition and Understanding, 2003. ASRU '03*, pages 135–140, 2003.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

Paul Taylor. Hidden Markov Models for grapheme to phoneme conversion. In *Proceedings of the 9th European Conference on Speech Communication and Technology*, 2005.

Kristina Toutanova and Robert C. Moore. Pronunciation modeling for improved spelling correction. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151, 2001.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004. ISBN 1-58113-828-5.

Raghavendra Udupa, K. Saravanan, A. Kumaran, and Jagadeesh Jagarlamudi. Mining named entity transliteration equivalents from comparable corpora. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1423–1424. ACM, 2008. ISBN 978-1-59593-991-3.

Antal van den Bosch. *Learning to pronounce written words: A study in inductive language learning*. PhD thesis, Universiteit Maastricht, 1997.

Gabriel Webster. Improving letter-to-pronunciation accuracy with automatic morphologically-based stress prediction. In *ICSLP*, pages 2573–2576, 2004.

Dmitry Zelenko and Chinatsu Aone. Discriminative methods for transliteration. In *EMNLP*, pages 612–617, 2006.

Richard Zens and Hermann Ney. Improvements in phrase-based statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 257–264, 2004.

Yue Zhang and Stephen Clark. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896. Association for Computational Linguistics, 2008.

Zhi-Hua Zhou and Min-Ling Zhang. Multi-instance multi-label learning with application to scene classification. In *NIPS*, pages 1609–1616, 2006.