Multilingual Spoken Term Detection: Finding and Testing New Pronunciations

Richard Sproat, Jim Baker, Martin Jansche, Bhuvana Ramabhadran, Michael Riley, Murat Saraçlar, Abhinav Sethy, Patrick Wolfe, Sanjeev Khudanpur, Arnab Ghoshal, Kristy Hollingshead, Chris White, Ting Qian, Erica Cooper, Morgan Ulinski

©2008, The Authors
## Contents

1 Abstract .................................................. 1

2 Web-Derived Pronunciations .................................. 3
   2.1 Introduction ........................................ 3
   2.2 Pronunciation Extraction ..................................... 3
      2.2.1 IPA Pronunciation Extraction ......................... 4
      2.2.2 Ad-hoc Pronunciation Extraction ..................... 4
   2.3 Pronunciation Extraction Validation ......................... 4
   2.4 Pronunciation Normalization ................................ 5
      2.4.1 IPA Pronunciation Normalization ..................... 6
      2.4.2 Ad-hoc Pronunciation Normalization ................. 8
   2.5 Conclusion ........................................... 9

3 Interscriptal Transcription .................................. 11
   3.1 Introduction ........................................ 11
   3.2 Previous Work ....................................... 13
   3.3 Overview of deliverables ................................ 15
   3.4 String Distances for Interscriptal Transcription ............ 15
      3.4.1 Framework ....................................... 16
         3.4.1.1 Data ........................................ 16
         3.4.1.2 Language Representation ....................... 16
         3.4.1.3 Discriminative Modeling Using Perceptron ......... 17
      3.4.2 Experimental Results .............................. 18
         3.4.2.1 Accuracy of Models for English-Chinese Transcription ... 19
         3.4.2.2 Effect of Training Size ........................ 19
         3.4.2.3 Using Negative Neighbors as Negative Examples .... 21
      3.4.3 Extensibility ..................................... 21
   3.5 Score Combination for Matching Transliteration Pairs ......... 23
      3.5.1 Introduction ...................................... 23
      3.5.2 Methodology ...................................... 23
         3.5.2.1 Orthographic Match ........................... 24
         3.5.2.2 Phonetic Match ................................ 24
         3.5.2.3 Temporal Distributional Similarity ............... 25
         3.5.2.4 Perceptron Classifier ............................ 25
         3.5.2.5 Score Combination .............................. 26
      3.5.3 Experimental Setup ................................ 27
      3.5.4 Results: Ranking .................................. 27
      3.5.5 Results: Classification ............................ 28
      3.5.6 Discussion & Future Work ........................... 29
   3.6 ScriptTranscriber ...................................... 29
   3.7 Ongoing Work ......................................... 34
4 Multilingual Spoken Term Detection

4.1 Part I - Effect of Pronunciations on OOV Queries in Spoken Term Detection

4.1.1 Introduction

4.1.2 Methods

4.1.2.1 WFST-based Spoken Term Detection

4.1.2.2 Query Forming and Expansion for Phonetic Search

4.1.3 Experiments

4.1.3.1 Experimental Setup

4.1.3.2 Results

4.1.4 Pronunciations derived from the web

4.1.5 Use of a foreign language: Turkish

4.1.6 Part I - Summary

4.2 Part II - Unsupervised pronunciation validation

4.2.1 Introduction

4.2.1.1 Baseline Supervised Method

4.2.1.2 Data set, OOV terms, systems

4.2.1.3 Supervised validation

4.2.2 Unsupervised Method

4.2.3 Results

4.2.3.1 Large vocabulary continuous speech recognition

4.2.3.2 Spoken term detection

4.2.4 Part II - Summary

4.3 Conclusion

5 Acknowledgments
## List of Figures

2.1 Precision vs. recall in pronunciation extraction validation. ........................................ 5
2.2 Cross-validation results by website. ............................................................................. 7
2.3 Effect of pronunciation normalization – L2P models trained using normalized web data are better at predicting the reference pronunciations. .................................................. 8
2.4 Performance on rare words – normalized web pronunciations help a lot on rare words. .... 9

3.1 Some automatically found n-tuples of transcriptions mined from the Web. .................. 12
3.2 Some automatically found pairs of transcriptions mined from the Web. ....................... 13
3.3 Temporal cooccurrence of the placenames Nunavut and Aglukkaq in English and Inuktitut in the Nunavut Hansards. ................................................................. 14
3.4 In the model of English letters and Chinese phonemic transcription (Worldbet), performance increases as a function of training size. ......................................................... 20
3.5 Increase of training size correlates with increase of performance in general. ................. 20
3.6 Model trained with negative neighbors performed poorly. ........................................ 21
3.7 Discriminating English-Korean transcription pairs using English letters and Worldbet symbols for Korean. ................................................................. 22
3.8 Discriminating English-Korean transcription pairs using English letters and Worldbet symbols for Arabic. ............................................................. 22
3.9 Examples of English/foreign transcription pairs, shown here with their pronunciations in each language. ............................................................ 24
3.10 Examples of the distributional similarity of the occurrences over time of English terms with their transcription pairs in both Chinese and Arabic. " Indicates a correct pairing. ................. 26
3.11 Examples of the best-matched transcription pairs according to the phonetic-match approach and a temporal correlation–match approach. Only the term marked by an " is a correct match. .... 27
3.12 XML document structure for ScriptTranscriber. ...................................................... 30
3.13 Interactive use of the ScriptTranscriber tools. .......................................................... 34

4.1 Index structures ........................................................................................................... 36
4.2 Comparison of 1-pass & 2-pass strategies in terms of retrieval performance and runtime ... 38
4.3 ATWV vs N-best L2S Pronunciations ........................................................................... 40
4.4 Combined DET plot for weighted L2S pronunciations ................................................. 40
4.5 English results: Pairwise union of Pronunciations ....................................................... 42
4.6 English results: Subword indexing using various pronunciations ................................. 43
4.7 DET curve on Turkish audio ......................................................................................... 43
4.8 Phone Error Rate w.r.t. reference lexicon ................................................................. 47
4.9 STD DET Curves ........................................................................................................ 47
List of Tables

2.1 Ad-hoc pronunciation extraction patterns and counts. ........................................... 4
2.2 Phoneme error rates (in %) for 5-fold cross validation on the intersection between Pronlex and Web-IPA pronunciations. ................................................................. 6

3.1 Examples of features and associated costs. Pseudofeatures are shown in boldface. Exceptional denotes a situation such as the semivowel [j] substituting for the affricate [dZ]. Substitutions between these two sounds actually occur frequently in second-language error data. 12
3.2 Worldbet as a phonetic representation of Chinese characters. ................................. 17
3.3 Model accuracy on positive and negative test cases, and its overall performance, compared to the model based on phonetic features as well as the baseline (a blind model). 19
3.4 Results on ranking English-Chinese or English-Arabic transcription pairs, mined from comparable corpora. Top three rows are single-score results; bottom four rows are linear-combination results. MRR: Mean reciprocal-rank. ................................................................. 28
3.5 True-positive matrices on classifying English-Chinese or English-Arabic transcription pairs as either valid (“positive”) or invalid (“negative”) pairs. .................................................. 29

4.1 Reflex Results ........................................................................................................ 39
4.2 Best Performing N-best L2S Pronunciations .......................................................... 40
4.3 S2S N-best Pronunciations expanding L2S output .................................................. 41
4.4 Number of OOVs that had pronunciations on the web ........................................... 41
4.5 English results:Analysis of L2S v/s Web-based pronunciations with positive impact 42
4.6 English results:Analysis of L2S v/s Web-based pronunciations with negative impact 42
4.7 Example pronunciations and PER ........................................................................ 46
4.8 LVCSR WER ........................................................................................................ 47
Chapter 1

Abstract

When you listen to the evening news, or read a newspaper, book or web site, there is a good chance that you will hear or see a term — perhaps a name, perhaps a technical term — that you have never seen before. Such words are often novel or rare and are often names (of people, places, organizations...). They are hard for humans to process, but they are even harder for automatic speech and language processing systems.

For a single language, a speech recognition or text-to-speech system needs to know how to pronounce a word to recognize or say it. For two languages, in particular a pair with different writing systems, a search engine or document summarizer needs to know how to transcribe one word to another to retrieve or distill across languages. For example, the soccer player written in English text as Maciej Zurawski would appear as 마치에이 주라브스키 (ma-chi-e-i ju-ra-beu-seu-ki) in Korean

In this project we attack both problems – unusual term pronunciation and term transcription. For pronunciation, we make use of the huge numbers of pronunciations that are now available in various forms on the web to mine pronunciations. This ranges from straightforward, such as dictionary sites and Wikipedia entries where people use a fairly strict phonetic transcription system such as IPA, to difficult such as:

- Trio Shares Nobel Prize in Medicine – The Nobel is a particularly striking achievement for Capecchi (pronounced kuh-PEK’-ee).

Here we need to look in the vicinity of the name “Capecchi” to find the pronunciation, make use of the word “pronounced”, and then interpret the writer’s attempt to render the pronunciation using an English-based ad-hoc “phonetic” orthography. The problem is therefore one of entity extraction, where the entities to extract can be either relatively easy or relatively hard. A relatively easy case is Wikipedia, which uses standard IPA transcriptions that are clearly delimited by markup. On general web pages, tokens with Unicode IPA characters are potential pronunciations. Data extracted from Wikipedia can be matched against these tokens to provide training material for entity extraction. Statistical entity extractors for the more difficult case of ad-hoc phonetic transcriptions (such as “kuh-PEK’-ee” above) can be bootstrapped from unannotated web pages containing patterns such as “pronounced as”. These entity extractors make use of both the textual environment and the letter-to-sound constraints between the candidate pronunciation and its corresponding orthography.

We also use speech data to test possible pronunciation variants by comparing the performance of spoken term detection systems using these different variants. Pronunciations mined from the web are used to suggest pronunciations for spoken term detection; transcription are used to suggest reasonable candidates to search for in a speech stream in another language. We use a novel technique called delayed-decision testing to test candidate pronunciations in speech, and to choose the best one from a set of candidates via a sequential testing procedure, with the associated null hypothesis stating that all candidate pronunciations exhibit the same performance on average. Spoken term detection are in turn used for automatic labeling of practice data acquired to test this null hypothesis; however, this automatic labeling procedure inevitably induces false alarms as well as correct detections. Delayed-decision testing are then used to choose the correct pronunciation in spite of these false alarms, leading to improved pronunciations for newly identified terms.

For transcription, we use available resources – dictionaries, and text corpora – as well as methods for phonetic matching across scripts and tracking names across time in comparable corpora (such as news sources).
In previous work at UIUC, JHU and many other sites, people have investigated phonetic transcription models trained from lexicons. More recently, we have developed techniques to guess transcription equivalents using pronunciation estimates for English terms, pronunciation guesses for the foreign term, and phonetic distances based upon standard phonetic features as well as “pseudofeatures” based on phonetic substitutions observed in second-language learners of English. Reasonable transcription matches can be found using hand-tuned costs based on these features, though improved performance can be demonstrated by discriminative training of the weights on even a short dictionary of transcriptions. We have also investigated using time correlations of terms across comparable corpora, such as newswire text. Related terms, including transcriptions of the same name, distribute similarly in time, and this is powerful additional evidence over and above phonetic similarity.
Chapter 2
Web-Derived Pronunciations

2.1 Introduction

Knowing how to pronounce a word is important for automatic speech recognition and synthesis. Previous approaches have either employed trained persons to manually generate pronunciations, or have used letter-to-phoneme (L2P) rules, which were either hand-crafted or machine-learned from a manually transcribed corpus Elovitz et al. (1976); Dietterich (2002). The first approach is expensive, the second can be of variable quality, depending on the skill of the experts or size and quality of the transcribed data. We investigate a novel strategy of mining the huge quantities of pronunciation information on the Web.

Two kinds of pronunciations are common on the Web: The first is expressed in the International Phonetic Alphabet (IPA), for example ‘Lorraine Albright /Ol brəIt/’. IPA pronunciations use special symbols, such as ‘O’, which can unambiguously denote a particular English phoneme. However, there are no universally accepted conventions for transcribing pronunciations in IPA, and the use of IPA requires some skill. It is then not surprising that we find considerable variation in IPA strings captured on the Web and there is a need to normalize them to follow a common set of conventions.

The second, and more frequent, kind of pronunciations use an ad-hoc transcription based on a simpler or less ambiguous spelling than standard English orthography. For example, when we see ‘bruschetta (pronounced broo-SKET-uh)’, the intended pronunciation is more intuitively represented by the letters ‘SKET’ than it is by ‘schet’. Ad-hoc transcriptions follow the rules of English orthography and do not require any specialized skills. However, they do not provide a phonemic transcription, so one of our tasks is to predict phonemes from a combination of the standard orthography and ad-hoc transcription of a word.

Processing IPA and ad-hoc transcriptions proceeds in three major phases. In the extraction phase (Sec. 2.2) we find a candidate pronunciation and its corresponding orthographic form on a web page. In the second phase, extraction validation (Sec. 2.3), we determine if an orthography/pronunciation pair was correctly extracted. For example, most instances of ‘pronounced dead’ do not correspond pronunciations that we would like to keep. In the final normalization phase (Sec. 2.4), we canonicalize irregularities in the IPA pronunciations and map the ad-hoc pronunciations to their phonemic form.

2.2 Pronunciation Extraction

The extraction, validation and normalization steps used in this paper require letter-to-phoneme, letter-to-letter, or phoneme-to-phoneme models. Methods for constructing such models include those based on decision trees Black et al. (1998), pronunciation-by-analogy Marchand and Damper (2000), and hidden Markov models Taylor (2005). We chose to use n-gram models over pairs Bisani and Ney (2002).

For a letter-to-phoneme n-gram model, each orthographic and phonemic training example is first aligned, as in e.g. (w, w) (i, i) (m, m) (b, b) (a, a) (l, l) (e, e) (d, d) (n, n). Alignments are derived by training a unigram model of (letter, phoneme) pairs (including letter deletions and phoneme insertions) using EM from a flat start and subsequently finding the most likely sequence of pairs under the model. Each (letter, phoneme) pair is then treated as a single token for a Kneser-Ney n-gram model Kneser and Ney (1995).
Once built, the n-gram model is represented as a weighted finite-state transducer (FST), mapping letters to phonemes, using the OpenFst Library Allauzen et al. (2007), which allows easy implementation of the operations that follow.

Our pronunciations are extracted from Google’s web and news page repositories. The pages are restricted to those that Google has classified as in English and from non-EU countries. The extraction of IPA and of ad-hoc pronunciations uses different techniques.

### 2.2.1 IPA Pronunciation Extraction

The Unicode representation of most English words in IPA requires characters outside the ASCII range. For instance, only 3.8% to 8.6% (depending on transcription conventions) of the words in the 100K Pronlex dictionary\(^1\) have completely ASCII-representable IPA pronunciations (e.g., ‘beet’ /bit/). Most of the non-ASCII characters are drawn from the Unicode IPA extension range (0250–02AF), which are easily identified on web pages. Our candidate IPA pronunciations consist of web terms\(^2\) that are composed entirely of legal English IPA characters, that have at least one non-ASCII character, and that are delimited by a pair of forward slashes (’/ . . . /’), back slashes (’n . . . n’), or square brackets (’[. . . ]’).

Once these candidate IPA pronunciations are identified, the corresponding orthographic terms are next sought. To do so, an English phoneme-to-letter model, \(Pr[\lambda | \pi]\), which estimates the probability that an orthographic string \(\lambda\) corresponds to a given phonemic string \(\pi\), is used. First, a unigram letter-phoneme joint model, \(Pr[\lambda, \pi]\), is trained on the Pronlex dictionary using the method described above. We use a unigram model both to ensure wide generalization and to make it likely that the subsequent results do not depend greatly on the bootstrap English dictionary. With this model in hand, we extract that contiguous sequence of terms \(\lambda\), among the preceding twenty terms to each candidate pronunciation \(\pi\), that maximizes \(Pr[\lambda | \pi] = Pr[\lambda, \pi]/\Sigma_{\lambda}Pr[\lambda, \pi]\). We found 2.53M candidate orthographic and phonemic string pairs (309K unique pairs) in this way. These are then passed to extraction validation in Sec. 2.3.

### 2.2.2 Ad-hoc Pronunciation Extraction

Ad-hoc pronunciations are identified by matches to the regular expressions indicated in Table 2.1. To find the corresponding conventionally-spelled terms, an English letter-to-letter model, \(Pr[\lambda_2 | \lambda_1]\), which estimates the probability that the conventionally-spelled string \(\lambda_2\) corresponds to a given ad-hoc pronunciation string \(\lambda_1\), is used. Assuming that \(\lambda_1\) and \(\lambda_2\) are independent given their underlying phonemic pronunciation \(\pi\), \(Pr[\lambda_2 | \lambda_1] = \Sigma_{\pi}Pr[\lambda_2 | \pi]Pr[\pi | \lambda_1]\) (implemented by weighted FST composition). Given the unigram model \(Pr_u[\lambda, \pi]\) of Sec. 2.2.1, the estimates \(Pr[\lambda_2 | \pi] = Pr_u[\lambda_2, \pi]/\Sigma_{\lambda}Pr_u[\lambda, \pi]\) and \(Pr[\pi | \lambda_1] = Pr_u[\lambda_1, \pi]/\Sigma_{\pi}Pr_u[\lambda_1, \pi]\) are used.

We then extract that contiguous sequence of terms \(\lambda_2\), among the preceding eight terms to each candidate pronunciation \(\lambda_1\), that maximizes \(Pr[\lambda_2 | \lambda_1]\). We found 4.52M candidate orthographic and phonemic string pairs (568K unique pairs) with pair counts for specific patterns indicated in Table 2.1. These pairs are then passed to extraction validation described in the next section.

### 2.3 Pronunciation Extraction Validation

Once extraction has taken place, a validation step is applied to judge whether the items extracted are correct in the sense that they find each orthographic term and the corresponding pronunciation provided word-for-word. We began by annotating 667 randomly-selected (orthography, IPA pronunciation) pairs and

<table>
<thead>
<tr>
<th>Type</th>
<th>Pattern</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>paren</td>
<td><code>(pronounced (as |like )?((|)[\^)]\+)\))</code></td>
<td>3415K</td>
</tr>
<tr>
<td>quote</td>
<td>pronounced (as |like )?&quot;((&quot;)+)&quot;</td>
<td>835K</td>
</tr>
<tr>
<td>comma</td>
<td>, pronounced (as |like )?([^]+)</td>
<td>267K</td>
</tr>
</tbody>
</table>

\(^1\)CALLHOME American English Lexicon, LDC97L20.

\(^2\)By terms we mean tokens exclusive of punctuation and HTML markup.
1000 (orthography, ad-hoc pronunciation) pairs. We use some of this as classifier training data and some for extraction evaluation.

Sixteen features of the IPA pronunciations and 57 features of the ad-hoc pronunciations are computed for this data. Features shared among both types of pronunciations include the string length of the extracted orthography and pronunciation, the distance between them, the presence of certain substrings (e.g. spaces, function words, non-alphabetic characters), and the log probabilities assigned by the alignment models used during the extraction. For the IPA pronunciations, the extracted pronunciation is aligned with the expected pronunciation – predicted from the extracted orthography by a 5-gram model trained on Pronlex – and per-phoneme alignment features are computed. These include the fraction of mismatched consonants and vowels, since we noticed that vowel mismatches are common in good extractions but consonant mismatches are highly indicative of bad extractions. Additional features for the ad-hoc pronunciations include letter-to-letter log probabilities, from n-gram pair models ranging from unigram to trigram, counts of insertions and deletions in the best alignment, and capitalization styles, since these often signal bad extractions.

Support vector machine classifiers were constructed separately for the IPA and ad-hoc pronunciation data using these features. Five-fold cross-validation was used to produce the precision-recall curves in Fig. 2.1, parameterized by the SVM-generated scores. In particular, the IPA extraction classifier has a precision of 96.2% when the recall was 88.2%, while the ad-hoc classifier has a precision of 98.1% when the recall was 87.5% (indicated by dots in Fig. 2.1).

To summarize, our extraction consists of a simple first-pass extraction step, suitable for efficiently analyzing a large number of web pages, followed by a more comprehensive validation step that has high precision with good recall. Given this high recall and the fact that most extraction errors, in our error analysis of a subsample, have no correct alternatives on the given page, we feel confident about this two-step approach.

### 2.4 Pronunciation Normalization

Up to this point, we have extracted millions of candidate IPA and ad-hoc pronunciations from the Web with high precision. We refer to the collection of extracted and validated data as the Web-IPA lexicon and the ad-hoc lexicon. The Web-IPA lexicon is based on extractions from websites that use idiosyncratic conventions (see below), while the ad-hoc pronunciations are still in an orthographic form. In both cases, they need to be normalized to a standard phonemic form to be useful for many applications.

Our training and test data are based on a subset of words in the web-derived data whose orthographies also occur in Pronlex. By using only the set of words that appear in both the lexica, we eliminate any overall sampling bias in either of the lexica, and focus solely on the pronunciations. We use a 97K word subset of the Web-IPA lexicon for these experiments, which has 30K words in common with Pronlex, with an average of 1.07 pronunciations per word in Pronlex, and 1.87 pronunciations per word in Web-IPA. In the next subsection, we also consider smaller subsets of this dataset that were derived by a similar methodology. The
training data for ad-hoc normalization was augmented by words whose pronunciations could be assembled from hyphenated portions of the ad-hoc transcription (e.g. if the ad-hoc transcription of ‘Circe’ is ‘Sir-see’, we look up the Pronlex phonemic transcriptions of ‘sir’ and ‘see’).

Some of our test sets, further described below, are drawn at random from the 30K Pronlex/Web-IPA lexicon. For others, we set aside rare words as test data, chosen by low counts in the HUB4 Broadcast News corpora. We are interested in rare words because they are less likely to occur in existing lexica. Handling these otherwise out-of-vocabulary (OOV) words is important in many applications.

We evaluate pronunciations by aligning a predicted phoneme string with a reference and computing the phoneme error rate (PhER) – analogous to word error rate in automatic speech recognition – as the number of insertions, deletions, and substitutions divided by the number of phonemes in the reference (times 100%). In cases of multiple predicted or reference pronunciations, the pair with the lowest PhER is chosen.

2.4.1 IPA Pronunciation Normalization

We first compare the quality of the Web-IPA lexicon with Pronlex, by performing 5-fold cross-validation experiments on their orthographic intersection described above. For each cross-validation run, two L2P models are trained on the same 24K subset of the intersection – one using the Pronlex pronunciations, and the other using the Web-IPA pronunciations. Each of the models is then used to generate candidate pronunciations for the same 6K subset left out of the training. The two sets of generated candidate pronunciations are then scored against the test pronunciations from both lexica, giving us four PhER numbers. The overall PhER for these four cases are shown in Table 2.2.

<table>
<thead>
<tr>
<th></th>
<th>Test</th>
<th>Train</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pronlex</td>
<td>6.35</td>
<td>14.33</td>
</tr>
<tr>
<td>Web-IPA</td>
<td>17.10</td>
<td>12.98</td>
</tr>
</tbody>
</table>

Table 2.2: Phoneme error rates (in %) for 5-fold cross validation on the intersection between Pronlex and Web-IPA pronunciations.

At a first glance the PhER numbers presented in Table 2.2 may suggest that the Web-IPA data is inherently of lower quality. But a very different picture emerges when one breaks down these PhER numbers by individual websites. We repeated the above cross-validation experiments, but instead using the entire Web-IPA data, the orthographic intersection was done using data collected from individual websites. Fig. 2.2 shows the same four PhER numbers for 7 of the 10 websites with the most extracted pronunciations. We notice that for several websites, L2P models trained using the web pronunciations are almost as good at predicting the website pronunciations (red bars) as a model trained on Pronlex is at predicting Pronlex pronunciations (dark blue bars). However, in all cases, models trained on web data are poor predictors of Pronlex data, and vice versa.

These experiments demonstrate that websites vary in the quality of pronunciations available from them. Moreover, the websites list different pronunciations than what one would obtain from Pronlex. The differences can be caused both by improper use of IPA symbols, as well as other site-specific conventions. For instance, ‘graduate’ is pronounced as either /ɡrædʒuət/ or /ɡrædʒuət/ in Pronlex, but appears as /ɡɹədʒuət/, /ɡrædʒuət/, /ɡrædʒuət/, and /ɡrædʒuət/ among the ten most frequent websites.

**Site-specific normalization**  The considerable variability of pronunciations across websites strongly motivate the need for a site-specific normalization of the pronunciations to a more site-neutral target form. Here we use Pronlex as our target. As before, we find the orthographic intersection of the lexicon obtained from a website and Pronlex. If multiple pronunciations are present, then the two with the smallest phoneme edit distance are selected. Using these pronunciation pairs we train a phoneme-to-phoneme (P2P) transduction model, which takes a pronunciation obtained from the website and converts it to a Pronlex-like form.

The model for the P2P normalizing transducer is identical to the L2P models described earlier, the only difference being that the P2P models are trained on aligned (phoneme, phoneme) pairs, instead of (letter, phoneme) pairs. For the cross-validation experiments, the normalizing transducer is trained on the
pronunciation pairs collected from the two training lexica, and is then used to normalize the pronunciations in the training lexicon obtained from the website. An L2P model trained on the normalized pronunciations is then used to generate candidate pronunciations for the test set words, which are scored against their Pronlex references.

The P2P transducers are trained using varying $n$-gram orders, and the results are presented in Fig. 2.3. As one can clearly see, normalization helps to improve the quality of the pronunciations obtained from the web. Notice in particular that the normalized pronunciations generated by a 5-gram model (light tan bars) have a PhER that’s comparable to the pronunciations predicted by a model trained on Pronlex (dark blue bars). Based on this, we conclude that L2P models trained on normalized Web-IPA pronunciations are as good as models trained on comparable amounts of Pronlex.

**Performance on rare words** To test performance on rare words, we remove from Pronlex any word with a frequency of less than 2 in the Broadcast News (BN) corpus. Among these rare words, the ones that are found in the extracted Web-IPA lexicon form our test set (about 3.8K words). Moreover, while creating a hand-built lexicon, it is natural to annotate the most frequent words. To replicate this we subdivide the Pronlex words, with BN frequency of at least 2, into 5 subsets based on decreasing frequency — the first one contains 20% of the most frequent words, the second one 40%, third with 60%, fourth 80%, and the fifth 100%.

For each of the subsets of Pronlex, we generate candidate pronunciations for the words in our rare-word test set using each of the following three methods:

1. An L2P model is trained on the subset of Pronlex, and then used to generate pronunciations for the rare words.

2. Pronunciations from the 10 most frequent websites are normalized using only the subset of Pronlex,
Figure 2.3: Effect of pronunciation normalization – L2P models trained using normalized web data are better at predicting the reference pronunciations.

and are then pooled together. Rare words are then looked up in this lexicon.

3. The normalized Web-IPA and the Pronlex subset are combined together and used to train another L2P model, which is then used to generate the pronunciations.

Fig. 2.4 shows the PhER on the rare words using each of the three methods described above, for varying amounts of Pronlex data used. The normalized Web-IPA data clearly produces better pronunciations for rare words. Of particular interest is the fact that the Web-IPA, when normalized using only 20% of the hand-crafted Pronlex dictionary (roughly 10K most frequent words), already produces pronunciations that are as good as those generated by an L2P model trained on the whole of Pronlex.

2.4.2 Ad-hoc Pronunciation Normalization

For ad-hoc normalization our task is to predict phonemic transcriptions from the extracted ad-hoc transcriptions, which are in orthographic form, but presumably reveal the intended pronunciation of a word more easily than the standard orthography. We investigated four ways of predicting phonemes from the extracted (orthography, ad-hoc transcription) pairs: (1) Apply a letter-to-phoneme model to the standard orthography (a competitive baseline). (2) Apply a letter-to-phoneme model to the ad-hoc transcription. (3) Model the phonemes as the latent source in a noisy channel model with independent channels for the orthography and ad-hoc transcription. (4) Train a language model on aligned (orthography, ad-hoc, phoneme) triples and apply it to the orthography and ad-hoc transcription.

We evaluate the predicted phoneme strings on a test set with 256 words that are associated with extracted ad-hoc and phonemic pronunciations manually transcribed to correspond both to the orthographic and ad-hoc forms. This yielded a total of 1181 phonemes.
For (1) we trained a 5-gram L2P model on a subset of Pronlex from which the test vocabulary was removed, achieving 29.5% PhER. Next (2) we trained a 5-gram L2P model on the 43K word training dictionary described earlier. This ignores the orthography and predicts phonemes directly from ad-hoc transcription, giving 20.5% PhER.

By contrast, the remaining two models use both the orthography and ad-hoc transcription to predict the phonemes. Model (3) is the noisy channel model $\Pr[\lambda_1, \lambda_2, \pi] = \Pr[\lambda_1 | \pi] \Pr[\lambda_2 | \pi] \Pr[\pi]$ which generates a latent pronunciation $\pi$ and, conditional on $\pi$, generates the orthography $\lambda_1$ and ad-hoc transcription $\lambda_2$ independently. It can be implemented straightforwardly in terms of the joint and conditional transducer models discussed in Sec. 2.2. This achieves 19.4% PhER. The last model (4) drops the independence assumption. It is a 5-gram language model on (orthography, ad-hoc, phoneme) triples, trained on the 43K lexicon by first aligning ad-hoc transcriptions with phonemes and then aligning the orthography with the already aligned (ad-hoc, phoneme) pairs. During testing the model is first combined with the orthography to predict (ad-hoc, phoneme) pairs, and those are further combined with the observed ad-hoc transcription to predict the phonemes. This model achieves 18.8% PhER – a 36% relative error rate reduction over the baseline model (1). We conclude that ad-hoc pronunciations – alone or in combination with the standard orthography – are extremely useful for predicting the pronunciations of unseen rare words.

2.5 Conclusion

Large quantities of human-supplied pronunciations are available on the Web, which we exploit to build pronunciation lexica comparable in quality to Pronlex and larger in size. Our methods yield more than 7M occurrences of raw English pronunciations (IPA plus ad-hoc), which we will make available to the public. To our knowledge, this is the first study of its kind; we are aware of related work Sumita and Sugaya (2006), which addresses a more restricted problem for Japanese by exploiting its multiple writing systems. Our approach can be used to bootstrap pronunciation lexica for any language where IPA or similar resources are
available (preliminary work on French and German holds promise).

One issue that we did not address is the usefulness of a pronunciation. For example, ad-hoc transcriptions of common words often highlight unusual pronunciations (e.g. ‘cheenah’ for ‘China’, which is a Spanish first name). This would be scored correct in Sec. 2.4.2, but there is a question of how many of these rare pronunciations we would want to put in our lexicon.
Chapter 3

Interscriptal Transcription

3.1 Introduction

A key problem in multilingual named entity recognition is recognition of the same name spelled in different ways in different scripts. For example, the (Polish) soccer player Maciej Zurawski’s name appears as 마치에이 주라브스키 in Korean, マチエ・ジュラフスキー in Japanese, Мачей, Журавский in Russian, and Ματσεί Ζουραφσκι in Greek, and 马西耶·茹拉夫斯基 in Chinese. All of these transcriptions share one thing in common: they represent attempts to render, in the various scripts used by the languages in question, a pronunciation that matches the original as closely as possible. The notion of match here can be quite loose, as it depends upon at least three factors:

- Knowledge of the original “correct” pronunciation of the name, which may be faulty.
- How well the name fits into the phonotactics of the target language.
- Possible language-specific strategies for spelling foreign names.

At the outset we wish to settle a terminological issue. The phenomenon we have just illustrated is most commonly referred to as transliteration in the literature. However, following Halpern (2007), we suggest this usage is wrong. Properly construed, transliteration is a one-for-one — and hence unambiguously reversible — mapping between two symbol sets. Basically, true transliteration is a technical or scientific system for representing one script in terms of another: one such system is Buckwalter’s system for transliterating Arabic (Buckwalter 2002). What people typically call transliteration is really an instance of transcription. Transcription includes phonetic transcription, where a word is represented not in its standard orthography, but as a sequence of symbols in some alphabet (e.g., IPA, ArpaBet, WorldBet) that represents the pronunciation of the word; as well as what Halpern terms popular transcription, instances of which have been illustrated above, where someone adopts a more or less regular system for representing a name in a foreign script in one’s own script.

Henceforth, we will adopt the term transcription, or popular transcription or interscriptal transcription if that term is unclear in context.

Figure 3.1 shows examples of the kind of phenomenon we are talking about. For example, the first line contains spellings of the word guru in Tamil, Kannada, Roman, Cyrillic, Bangla, Malayalam and Devanagari scripts. The fourth line contains various spellings of the word hotel in Roman and Cyrillic. Note that some Cyrillic spellings transcribe the ‘h’ (e.g. хотен), reflecting normal English pronunciation; whereas others do not (e.g. отель), presumably reflecting French pronunciation. There are some errors: for example in the third line from the bottom, 브라운, БРАУН/Браун are transcriptions of Brown, but Брайан is evidently Brian. A larger set of transcription pairs mined from the web is shown in Figure 3.2. These pairs were found as follows. Using the Google infrastructure, we found strings that matched the pattern $c_1c_2...c_n(d_1d_2...d_m)$, where $c_i$ are elements of one script, and $d_i$ are elements of another, enclosed in parentheses. Frequently, such cases involve a text in one language, containing a word or name from another language, with the original form of the word/name being given in the source language. Obviously not all such strings involve such cases, and even when they do, there is the issue of which word(s) in the $c$’s correspond to the transcriptions among
To filter the latter, we used the hand-built phonetic match model described in Tao et al. (2006); Yoon et al. (2007), which uses both traditional articulatory/acoustic phonetic features, as well as ‘pseudo-features’ based on common errors made by second-language learners. (As an example of the latter, it is common for speakers of several source languages who are L2 speakers of English to substitute /dʃ/ for /j/, a change that is hard to describe in terms of simple phonetic feature substitutions. All mismatches in features are weighted. Costs of these feature substitutions are shown in Table 3.1, from Tao et al. (2006). This hand-built distance measure is built into the ScriptTranscriber toolkit that we describe in Section 3.6.

<table>
<thead>
<tr>
<th>Class</th>
<th>Feature</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major features and Consonant Del</td>
<td>consonant</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>sonorant</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>consonant deletion</strong></td>
<td></td>
</tr>
<tr>
<td>Place features and Vowel Del</td>
<td>coronal</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>vowel del/ins</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>stop/fricative consonant del at coda position</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>h del/ins</td>
<td></td>
</tr>
<tr>
<td>Manner features</td>
<td>nasal</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td><strong>dorsal feature for palatal consonants</strong></td>
<td></td>
</tr>
<tr>
<td>Vowel features and Exceptions</td>
<td>vowel height</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>vowel place</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>exceptional</strong></td>
<td></td>
</tr>
<tr>
<td>Manner/ Laryngeal features</td>
<td>continuous</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>voicing</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Examples of features and associated costs. Pseudofeatures are shown in boldface. Exceptional denotes a situation such as the semivowel [j] substituting for the affricate [dZ]. Substitutions between these two sounds actually occur frequently in second-language error data.

Transcription is important not only as a part of modern problems in information extraction and machine translation, but it has also played a key role in history. The fact that names were spelled phonetically in different scripts allowed for Thomas Young and Jean François Champollion’s initial work on the decipherment of Egyptian, Georg Friedrich Grotefend’s breakthroughs in the decipherment of Persian cuneiform, and Michael Ventris’ decipherment of Mycenaean Linear B. In the most famous of these, the decipherment of Egyptian, it was because the Egyptians phonetically transcribed in hieroglyphs the Greek names of their last rulers — thus ΠΤΟΛΕΜΑΙΟΣ ‘Ptolemy’ as ptwlmyys — that gave Young and Champollion their initial keys to the code.
3.2 Previous Work

In Knight and Graehl (1998) a source-channel model of transcription between English and Japanese was proposed. For much work in this field the problem of transcription can be broken into two subproblems. The first is the prediction of an appropriate transcription given a source term and a particular target language. For example, what is the most likely rendition of the name Barack Obama in Japanese, or Korean, or Arabic.

A second, often harder problem, is what is often called ‘backwards transliteration’: if I see 바락 오바마 in a Korean text and can identify it as coming from English, what is its most likely English source?

Other work on transcription since Knight and Graehl (1998) includes AbdulJaleel and Larkey (2002, 2003); Al-Onaizan and Knight (2002); Arbabi et al. (1994); Gao (2004); Gao et al. (2004); Goldwasser and Roth (2008); Hermajakob et al. (2008); Jiang et al. (2007); Kang and Kim (2000); Kawtrkul et al. (1998); Kuo et al. (2007); Larkey et al. (2003); Lee and Choi (1998); Li et al. (2007); Meng et al. (2001); Pouliquen (2008); Shao and Ng (2004); Sherif and Kondrak (2007); Virga and Khudanpur (2003); Yang et al. (2008).

In previous work leading up to this project Sproat et al. (2006); Tao et al. (2006); Klementiev and Roth (2006); Yoon et al. (2007) and the project itself we have been interested in a different problem namely finding transcription pairs (tuples) in comparable corpora — i.e. corpora that deal in roughly the same topics, but where the documents are not translations of each other. A typical example of a comparable corpus would be texts from a pair of newspapers in two or more languages. Assuming the two countries are equally connected with the rest of the world, one can expect that there will be an overlap in the topics they cover, and that therefore many of the same named entities will occur in both.

For example consider that in early September 2008, many people were focussed on Hurricane Gustav, and
what damage it might inflict upon the US oil industry in the Gulf of Mexico, or on the city of New Orleans, which three years previously had been devastated by Hurricane Katrina. Not surprisingly, there was much discussion of this topic in the international news and a Google search on the Chinese for *Gustav* — 古斯塔夫 — turned up many news stories on this topic in Chinese online sources. Here there was no bilingual text: it simply seemed likely that this topic would be newsworthy enough to appear in Chinese-language newspapers, which is indeed what happened.

Comparable corpora have been used a lot in work on translation and information extraction; see, inter alia, Fung (1995); Rapp (1995); Tanaka and Iwasaki (1996); Franz et al. (1998); Ballesteros and Croft (1998); Masuichi et al. (2000); Sadat et al. (2003); Tao and Zhai (2005). Work on transcription from comparable corpora includes Shao and Ng (2004); Sadat et al. (2003); Sproat et al. (2006); Tao et al. (2006); Klementiev and Roth (2006); Yoon et al. (2007).

String edit distances have of course also been studied widely in a variety of applications. The classic paper is Kruskal (1999). Some recent work on learning string-edit distance in various application domains include Ristad and Yianilos (1998); McCallum et al. (2005); Cohen et al. (2003); Takasu et al. (2007).

In our previous work most directly related to the work in this proposal Sproat et al. (2006); Tao et al. (2006); Klementiev and Roth (2006); Yoon et al. (2007), we used comparable corpora to evaluate various methods for mining transcriptions in English, Chinese, Korean, Arabic, Hindi and Russian. We considered hand-tuned phonetic distances (substitution costs) Tao et al. (2006) that included traditional phonetic features as well as features derived from common phonetic substitutions made by second language learners. And we also used these features with substitution costs trained using linear separators Yoon et al. (2007). The other similarity measure was similarity of distribution across time. We used both time-domain measures — Pearson’s correlation of normalized frequency Sproat et al. (2006) — and frequency-based measures — similarity of the fast Fourier transform of distributions Klementiev and Roth (2006). In comparable corpora we expect related terms to correlate in their distribution over time; see for example Figure 3.3, which plots the similarity of the placenames *Nunavut* and *Aglukkaq* in the English and Inuktitut versions of the Nunavut Hansards.

One evaluation measure that we introduced in this prior work from the information retrieval literature is *mean reciprocal rank* (MRR). In an ordered list of items retrieved for a query, the reciprocal rank of the result is defined as $\frac{1}{n}$, where $n$ is the rank of the correct answer in the list. The MRR is averaged over a set of query results. While some have questioned the validity of this measure in rating transcriptions, in an
intended application where one has a given term in one language, and a list of candidate terms in the other, it is clear that the better the transcription system, the higher will be the rank of the correct candidate in the list of possible matches.

Comparable corpora are, in fact, a good source of evidence for the hard job of backwards transliteration, described above. It is pretty hard on the basis of phonetic models alone to determine the correct original spelling of a word transcribed into another script. But as Knight and Graehl’s original work discussed, a language model for the source language can narrow the targets considerably. Comparable corpora are a source of temporally appropriate language models. While in principle Gustav in a contemporary Chinese newspaper could come from a variety of original source names, in the current context it is unlikely to be anything besides Gustav.

3.3 Overview of deliverables

In light of the previous work that has been done on this problem the focus of this workshop was on better understanding which features make the most sense for string comparison and for temporal cooccurrence. Specifically we will provide results that improve out understanding of which features to use for discriminative raining of string comparison between transcriptions. We will discuss feature combination methods for string– and temporal cooccurrence.

To anticipate our results on string-comparison features: we show with data from Chinese that phonetic matches can be better than orthographic matches, if you can predict the pronunciation with reasonable accuracy. We also show that a very compact feature set – the hand-designed set of features from Yoon et al. (2007) works almost as well as feature sets containing hundreds or thousands of features derived by methods similar to those of Klementiev and Roth (2006). This suggests that linguistically derived features do have a useful place in transcription between scripts.

One of the stated goals of this workshop was to provide a set of open-source tools for mining transcriptions from comparable corpora. We discuss our ScriptTranscriber toolkit in Section 3.6.

We note at the outset some of the data that we have been using in this project:

- Lexicons:
  - Large (approx. 71,000 entry) English/Chinese name lexicon from the Linguistic Data Consortium (http://www.ldc.upenn.edu).
  - 9,000 entry Korean/English lexicon from public source
  - 1,750 entry Arabic/English lexicon from New Mexico State University
  - Large transcription lexicon from multiple pairs of languages derived from the Web (>200K good entries)
  - 295K geographical names from the National Geospatial-Intelligence Agency (http://www1.nga.mil)

- Corpora, from the Linguistic Data Consortium, unless otherwise indicated:
  - Chinese, English, Arabic gigaword
  - ISI English/Chinese, English/Arabic found parallel corpora
  - Less Commonly Taught Languages (LCTL) Thai/English corpus
  - Nunavut Hansards (English/Inuktitut) (http://www.assembly.nu.ca/english/debates/index.html)

3.4 String Distances for Interscriptal Transcription

Graphical similarity can be used as an alternative metric to explicit phonetic features in judging the quality of automatically extracted transcription pairs. The idea is that for most languages, graphical features often
correlate with phonetic features, from which transcriptions are typically derived. Klementiev and Roth Klementiev and Roth (2006) applied a letter-based n-gram string comparison method to modeling transcription pairs between Russian and English. In their example, the English name Powell can be transcribed into Russian as Pauel, where (ow, au) is an instance of letter-bigram pair features. A discriminative model can then be trained to learn these pair features.

Although this approach obviates the process of hand crafting phonetic rules, and generally yields competing results, it is not applicable in languages that have distinctive orthographies. The Chinese transcription for Powell, for example, has only three characters: “鲍威尔”. The difference in length creates a problem for effective alignment between these two scripts. Additionally, written Chinese is even less of a phonetic description of the language. Thus, a pair feature between English and Chinese does not really indicate phonetic correlation, making direct alignment seem awkward. Similar problems exist for English-Arabic and English-Korean transcription models: in Arabic, vowels are omitted from the written script; in Hangul, the phonemic alphabet of Korean, “letters” are organized into characters that correspond to syllabic blocks. In language pairs where graphical features no longer correlate with phonetic features, a letter-based model like above will form incorrect correlations.

The theme of our work then, is to find a way through which the methods of graphical comparison can be applied to these three language pairs (i.e. English-Chinese, English-Arabic, and English-Korean). We are particularly interested in looking for the best graphical features to use for each language pair. Since transcribed words tend to reflect the pronunciations of words in the language that they are transcribed from, we adopted the intuitive strategy of phonemic transformation on Arabic, Chinese and Korean. In doing so, the resulting phonemic transcriptions of Arabic, Chinese and Korean will better align with English letters.

Due to the availability of large amount of Chinese data, we focused on English-Chinese bi-directional transcription. In what follows, we describe the framework of our discriminative transcription model, followed by experimental feature selection in English-Chinese transcription and corresponding results. Extensibility of our approach to other language pairs is discussed in the context of English-Arabic and English-Korean transcriptions. Results show that using phonemic transcriptions of languages with non-alphabetic scripts is a generic solution to all three cases.

### 3.4.1 Framework

The entire framework of our discriminative model is described in this section. We set the discussion in the context of English-Chinese transcription for the ease of illustration.

#### 3.4.1.1 Data

We based our work on three parallel dictionaries of named entities, among which the English-Chinese dictionary is the largest one with nearly 75,000 pairs of transcriptions. However, the original dictionary consists of a number of Japanese Kunyomi words. They are written in Chinese characters but are transcribed into English using Japanese pronunciations; therefore, there is no phonetic correlation between them at all. We manunally removed these Japanese Kunyomi names from the dictionary. The resulting dictionary contains 71,548 entries. What remains after filtering is then partitioned into three parts: the first 90% of the dictionary used as the training set, the remaining part divided into 5% heldout and 5% testing parts (about 3,580 examples each).

The other two dictionaries are considerably smaller in size. The English-Arabic dictionary has 1,750 entries, while the English-Korean one has 9,046 entries.

#### 3.4.1.2 Language Representation

We look for alternative representations of languages that can easily be aligned with each other, so that it is possible to maximally exploit phonetic information from such graphical alignments. A representation could plainly be the original script in which the language is written. Otherwise, phonemic transformation is typically used.

#### 3.4.1.2.1 Representation of English

For English, it is possible to construct a string-based language model directly on its orthography. At the same time, there are plenty of pronunciation modeling toolkits for
converting English words to corresponding phonemes (at the model’s best guess). In our work, both representations are tested in combination with other languages. English phonemes are based on the pronunciation model of Festival.

3.4.1.2.2 Pinyin-based Phonemic Representation of Chinese  Our initial approach was to take Chinese characters and convert them to Pinyin strings. Pinyin is a quasi-phonemic transcription system used for describing the sounds of Chinese characters. Pinyin strings are written in regular English alphabet, although the pronunciation follows a rather unique set of rules. For example, the word huntington, transcribed to “亨廷顿” in Chinese, would be now represented in the Pinyin string heng ting dun (spaces between Pinyin syllables turned out to be a bad addition when corporated in discriminative modeling). In the case of homophones, where a character may have multiple pronunciations, the most frequent Pinyin string is always selected. This simple strategy actually makes sense: characters used in transcription should observe their most frequent pronunciation so that people are more likely to be able to pronounce the word.

Compared to characters, the Pinyin string of a Chinese word is more similar to the English counterpart in length. Pinyin also has the advantage of being an accurate and deterministic transformation scheme of Chinese characters. Given a character, it is always possible to map it onto the unique Pinyin string that represents its pronunciation (given our frequencist assumption). However, Pinyin strings are biased to the particular theory of Mandarin phonology that it is designed with, and therefore may ignore some phonetic differences that actually exist. For example, the fact that the three very different vowels in “si”, “shi” and “xi” are written with an “i” encodes the theory that at some level these are the same vowel, being influenced by the adjacent consonant. Other transcription systems, such as Wade-Giles, make different theoretical claims.

3.4.1.2.3 Worldbet-based Representation of Chinese  Worldbet Hieronymous (1993) attempts to represent all possible phones of human language in ASCII symbols. It has the merit of being independent of phonological theories of a particular language. Consequentially, it does not have the above-mentioned problem associated with Pinyin. In representation, Worldbet symbols are separated by spaces; a single symbol denotes a phone with one or more characters. Table 3.2 shows two examples in which English words and their transcriptions in Chinese characters, Pinyin, and Worldbet are listed separately.

<table>
<thead>
<tr>
<th>English</th>
<th>Character</th>
<th>Pinyin</th>
<th>Worldbet</th>
</tr>
</thead>
<tbody>
<tr>
<td>binci</td>
<td>宾奇</td>
<td>Bin Qi</td>
<td>p i n cCh i</td>
</tr>
<tr>
<td>basgil</td>
<td>巴什吉尔</td>
<td>Ba Shi Ji Er</td>
<td>p a sr &amp; n cC i &amp;r</td>
</tr>
</tbody>
</table>

Table 3.2: Worldbet as a phonetic representation of Chinese characters.

Note that the Worldbet strings are transcribed from Pinyin rather than directly from characters. Given Pinyin accurately and deterministically denotes the sound a character, a Pinyin-to-sound model has the advantage of reaching near perfect accuracy over other pronunciation models. On the other hand, Worldbet itself represents the sound of characters more directly, and thus fine-tunes the correlation between English and Chinese pairs. In fact, when we followed the same procedure (introduced later) to generate a feature map for English to Chinese Worldbet sequences as in the Pinyin-based model, using Worldbet gave twice as many features as found the Pinyin model.

3.4.1.2.4 Worldbet Representations for Arabic and Korean  Arabic and Korean are converted to the Worldbet-based representation using our toolkits.

3.4.1.3 Discriminative Modeling Using Perceptron

We used a single-layer perceptron for training a discriminative model to learn graphical correlations. Training examples and test cases are allowed to have feature vectors of different lengths. We first describe how features are derived from training data, and then briefly discuss the method for selecting optimal perceptron parameters.
3.4.1.3.1 Couplings of Bigram Substrings as Features

Given a parallel dictionary of two languages represented in appropriate forms, a set of letter-based bigram substrings are generated for each word in a transcription pair. Consider the previous example, where we have $w_E = huntington$ and $w_P = hengtingdun$ (Pinyin strings are concatenated together to resemble English orthography). Then, the sets of bigram substrings are:

$$w_E \Rightarrow \langle h, hu, \ldots, on, n_\rangle$$  
$$w_P \Rightarrow \langle h, he, \ldots, un, n_\rangle$$

Underscores in here indicate either the beginning or the end of a word. Couplings of substrings from both sets (such as, $< hu, he >$) are collected and used as training features of our model. We follow previous work where substrings that differ by one index position are paired Klementiev and Roth (2006). In addition to location-based restrictions, couplings that violate language-specific rules are also filtered out of the feature space. For example, Chinese do not allow consonant clusters; therefore, a coupling like $< nt, gt >$ offers a rather improbable interscriptal relation and is not considered. Finally, low-frequency features are excluded from the feature space (current threshold is 1).

Each feature that remains in the feature space after filtration is then given a unique ID. We compiled the training part of dictionary entries into positive examples, and created four times as many negative examples from positive ones with three different methods (see Experimental Results for details). Both categories of examples are represented in feature IDs for the perceptron to learn transcription patterns. Since there are more negative examples than positive ones, any model trained on such data will be biased towards negativity. However, in corpus-based automatic extraction of transcription pairs, only one or two will be correct among many candidates. Therefore, the bias desirably works in our favor.

3.4.1.3.2 Parametric Optimization

An iterative algorithm is applied to the training process to ensure that the best possible parameters are selected for the discriminative model. We are particularly concerned with two model parameters: feature cut-off value and the number of training iterations. The cut-off value determines how frequent a feature must appear in the training data to be actually considered. The number of iterations is most crucial because we want to prevent under-training as well as over-training.

In practice, we maintain an infinite training loop in which the values of these two parameters are increased by 1 in each step. At the end of each cycle, the current model’s performance is tested on the heldout set (5% of the dictionary size). That is, we monitor the performance of the model with respect to heldout data. If its performance stops improving for 3 consecutive cycles, we cease this iterative training process and report a set of optimal parameters. The detailed procedure is as follows:

- Set the model to run at cut-off values from 1 to 5 on the training data;
- For each feature cut-off value, train the model with 1 iteration;
- Increment the number of training iterations by one, retrain the model, and monitor the testing results on heldout data for the most recent three models;
- Repeat the above step; if the model stops improving by at least 0.04% in the 3-session interval, cease training, and report the first one of the most recent three models as the best one for current cut-off.

This iterative training process selects optimal values of cut-off and iteration number for a given training task in an ad-hoc fashion. We then use the particular model trained with optimal parameters in subsequent testing phases.

3.4.2 Experimental Results

Experimental results are reported in this section. Binary classification was carried out on the test data (5% of the dictionary) with models trained as described. Two classes of test cases are present in the test data: plausible transcription pairs (positive) and implausible ones (negative). These negative test cases are generated from positive ones as in training examples. We are primarily interested in finding how accurately the trained model can distinguish between these two kinds.
3.4.2.1 Accuracy of Models for English-Chinese Transcription

We experimented with three different models: the first one uses English letters and Chinese Pinyin strings (EL-CP); the second one uses English letters and Chinese phonemic transcription based on Worldbet (EL-CW); the third one uses English Phonemes produced by Festival instead of plain letters, but keeps the Chinese phonemic transcription in Worldbet (EP-CW). Additionally, we compared these models with a model based-on handcrafted phonetic features in the work of Yoon et al. (2007). With nearly 65,000 positive training examples and four times as many negative examples, these models all performed very well on the similarly proportioned test data. Table 3.3 shows the accuracy figures.

<table>
<thead>
<tr>
<th>Model</th>
<th>Positive</th>
<th>Negative</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonetic</td>
<td>88.5%</td>
<td>97.0%</td>
<td>95.5%</td>
</tr>
<tr>
<td>EL-CP</td>
<td>92.3%</td>
<td>98.1%</td>
<td>96.3%</td>
</tr>
<tr>
<td>EL-CW</td>
<td>97.7%</td>
<td>99.6%</td>
<td>99.4%</td>
</tr>
<tr>
<td>EP-CW</td>
<td>95.9%</td>
<td>99.1%</td>
<td>98.5%</td>
</tr>
<tr>
<td>Baseline</td>
<td>–</td>
<td>–</td>
<td>80.0%</td>
</tr>
</tbody>
</table>

Table 3.3: Model accuracy on positive and negative test cases, and its overall performance, compared to the model based on phonetic features as well as the baseline (a blind model).

All three models are more likely to correctly identify that a test case is an implausible transcription pair, than to identify it as plausible. This is not surprising, because, as we have stated, the ratio between the number of negative examples and that of positive ones is 4 to 1. Consequently, if a model were to classify every cases presented as negative, it would achieve a baseline accuracy of 80%. Fortunately, even the worse model above (English letters to Chinese Pinyin letters) still outperforms the baseline significantly by 16.3%.

Note that the best model in Table 3.3 is the English letters to Chinese Worldbet model. We think this is due to two obvious reasons. First, Worldbet, as discussed above, represents Mandarin pronunciation in a way that is unbiased by any phonological theories. Therefore, it is more faithful in capturing the phonetic properties of Mandarin words than Pinyin. Second, English letters are also better than phonemes because the latter may contain errors due to the pronunciation toolkit that we use. These noises debilitated the performance of the English phoneme to Chinese Worldbet model.

These data also indicate that discriminative training using perceptron is an effective method in the context of identifying English-Chinese transcription pairs. Compared to the method based on phonetic rules, it has a clear advantage due to a much larger set of available features. We show this method can be extend to building English-Arabic and English-Korean models in the next section.

3.4.2.2 Effect of Training Size

The above performance was reported from models with more than 300,000 examples in training data (including both positive and negative ones). To understand whether it is possible to achieve comparable performance with smaller training sets, we randomly sampled eight subsets from the training data, ranging from 500 to 250,000 examples. Trained models were tested on the same test data as used previously.

We evaluated the models for their accuracy on binary classification of positive and negative test cases using f-scores as in Formula 3.1.

\[ F = \frac{2 \times (\text{precision} \times \text{recall})}{\text{precision} + \text{recall}} \]  \hspace{1cm} (3.1)

That is, we evaluated the task of classifying test cases as if it were two information retrieval tasks. In one, the perceptron model tries to retrieve as many positive examples as possible (recall), while controlling for the number of false alarms (precision) at the same time. In the other, it tries to do the same thing with negative examples. But these two tasks are dependent since the classification targets are binary.

Figure 3.4 shows some interesting effects of training size on model performance. First of all, even with only 500 examples (100 positive and 400 negative), the trained model hits a decent benchmark – 92% accuracy for overall (however, positive f-score is apparently much worse than negative). As the training size increases,
overall accuracy improves quickly at first and then gradually reaches a plateau. Separately, positive and negatives scores are becoming less deviant and converging to the overall accuracy.

Accuracy as a function of training size appears to be a property of our model regardless of which language representation is selected. For example, in Figure 3.5, the left part is the same as Figure 3.4; the right part shows the curve that depicts the performance of the model using English phonemes. We can see it is increasing steeply at points where smaller training sets are, and gradually flattens towards the end. The trend tends to be very similar among all models that we have tried.

Figure 3.5: Increase of training size correlates with increase of performance in general.
3.4.2.3 Using Negative Neighbors as Negative Examples

Various methods for generating negative examples are also tried in our work. The most intuitive ones are 1) mismatching dictionary pairs so that each word in one language now corresponds to a random word in the other, and 2) permutating characters within words for one side of the dictionary. These two methods are effectively similar, since in both cases, the original graphical correlations are disrupted randomly. In fact, we observed very little changes when they were compared in experiments.

We came up with a third approach, in which negative examples of a transcription pair are derived by first numerating nearest neighbors of the word in one language, and then finding the corresponding transcriptions for these neighbors in the other language. When these transcriptions of neighbors are paired with the original word, they represent “hard cases” that should be considered implausible but whose pronunciations do sound similar.

A model trained with hard cases were expected to be able to not only perform as well as previous models, but also to detect minor details in given candidate pairs. Therefore, the negative neighbors should lead to a stronger model. However, when it was evaluated on the same test data, negative neighbors proved to be a source of confusion, as shown in Figure 3.6.

The model gradually regressed to a naive model (every test case is negative) with more training data. The regression was interesting because it implies that most correlational features represented in negative neighbors are in fact plausible transcribing patterns. As the model learns, it can correctly identify real negatives, but it also becomes more and more confused about which features are indicative of positiveness. In other words, positive features that exist in negative examples mislead the perceptron to recognize them as features of negativity.

3.4.3 Extensibility

The method that we used to build the discriminative model for English-Chinese transcription can be applied to any language pairs given the availability of a form of graphical representation. In particular, we extended it to English-Korean and English-Arabic transcriptions.

3.4.3.0.1 English-Korean Transcription  The English-Korean parallel dictionary contains 9,046 pairs of named entities in English and Korean. Similarly, we divided it into a training set, a heldout test, and a test set, and then applied the above procedures to the data. Specifically, the Hangul alphabet was converted to Worldbet symbols as well. Six subsets were drawn from the training data to study the effect of training size. Figure 3.7 shows an overview of results.
3.4.3.0.2 English-Arabic Transcription  The English-Arabic dictionary is even smaller, with only 1,750 pairs. In evaluating the data, we plotted the model’s performance for seven different training sizes with an equal difference of 1,000 examples (250 positive & 750 negative ones). Figure 3.8 shows the obtained results.

Overall, the discriminative framework for transcription seems a very robust model across different pairs of languages.

The decrease of performance at the tail of the curve showed that a discriminative model tends to be unstable when there is little training data. To build a truly useful English-Arabic transcription model, more data will be necessary.
3.5 Score Combination for Matching Transliteration Pairs

3.5.1 Introduction

In this work, we use knowledge from several sources to match pairs of terms from two languages that use different writing systems. The task of matching terms across languages is more complex than calculating a simple edit distance between two strings when the strings appear in different scripts. For example, the name of the 2004 Nobel Peace Prize winner written in English text as “Dr. Wangari Maathai” would appear as “旺加里马塔伊” in Chinese.

One common approach to this task is to phonetically match terms from one language against a list of terms from another language. This approach has been used fairly successfully in matching transcription pairs from several different language pairs, including English-Russian, English-Chinese, and English-Arabic (Klementiev and Roth, 2006; Sproat et al., 2006; Kuo et al., 2007). A related approach, and one which we do not explore here but leave to future work, is to generate new, plausible target-language spellings that preserve the sounds of the source-language name as much as possible. A third approach is to transform the two terms into a common script. For example, ‘马塔伊’ could be transformed using Chinese Pinyin characters into ‘ma ta yi’, and the Arabic transcription of Dr. Maathai’s name, ‘ماثائي’, would be transformed using the Buckwalter analysis into ‘mAvAy’. In Section 3.4, we demonstrated the success of comparing terms written in English to Chinese terms written using the Pinyin character set. Such an approach requires language-specific knowledge, but in many cases existing tools are available perform the transform. Another approach, quite different from the previous three which rely on information extracted from the orthography of a term, is to examine time correlations of terms across comparable corpora such as newswire text. Related terms tend to distribute similarly in time, so two terms with similar temporal distributions may be a valid transcription pair (Tao et al., 2006).

All of these approaches have been previously studied. In this work, however, we apply each approach to the same dataset in order to compare the results obtained by each. Furthermore, we look at results pairing English to Chinese transcription terms as well as results from the same dataset pairing English to Arabic transcription terms to explore whether there are any language-specific effects. We also strive to determine whether the different approaches provide orthogonal information. We will do this by combining the scores obtained by each approach. Previous work has shown linear combination to be useful; combining scores from a phonetic approach and a temporal correlation approach improved performance above each individual approach (Tao et al., 2006). We will explore a few different methods for score combination, and analyze some of the benefits and drawbacks of such combinations.

3.5.2 Methodology

Following Shao and Ng (2004); Klementiev and Roth (2006); Sproat et al. (2006), we mined comparable corpora for possible transcription pairs. Comparable corpora, such as news documents of the same period from different news agencies—and, for our purposes, in different scripts—are widely available. We will extract possible transcription pairs from this corpora, as described in Section 3.5.3, then score each pair of terms according to their orthographic, phonetic, and temporal correlation match. Note that we are extracting and matching possible pairs, rather than generating a transcription from each source term.

The benefit of using comparable corpora rather than parallel corpora, which consists of translated documents such as the parliamentary proceedings of the Canadian Hansards corpus and the European Parliamentary corpus, is that there is a fairly limited amount of parallel corpora available. On the other hand, data extracted from comparable corpora will be much “noisier.” There is no guarantee that the foreign side of the corpus contains a translation of the source document. Note that there is also no guarantee, with either comparable corpora or parallel corpora, that the correct transcription of the source term will be found in the foreign document. There are many reasons why a valid transcription may not be found, including the possibilities that: the source term is not a named entity and therefore should be translated rather than transcribed; the source term is a named entity but the correct transcription into another script is actually a combination of translation and transcription, which our methodology does not account for; or the correct transcription of the English named entity is simply not contained in the foreign document.

Thus our methodology will need to handle the overwhelming amount of data available from mining
<table>
<thead>
<tr>
<th>English term</th>
<th>Foreign term</th>
<th>English pronunciation</th>
<th>Foreign pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bondevik</td>
<td>邦德维克</td>
<td>b A: n d I v I k</td>
<td>p a N t &amp; v &amp; i kh &amp;</td>
</tr>
<tr>
<td>Canada</td>
<td>加拿大</td>
<td>k @ n &amp; d &amp;</td>
<td>c y a n a t a</td>
</tr>
<tr>
<td>Hilary</td>
<td>هيلاري</td>
<td>h I l &amp; r i;</td>
<td>h j l j r j</td>
</tr>
<tr>
<td>Palestine</td>
<td>فلسطين</td>
<td>p @ l &amp; s t a I n</td>
<td>f l s t ~ j n</td>
</tr>
</tbody>
</table>

Figure 3.9: Examples of English/foreign transcription pairs, shown here with their pronunciations in each language.

comparable corpora. We will find it useful to use heavy filtering techniques in order to reduce the number of transcription pairs to be analyzed. We also need to be mindful of the computational complexity required by any of our analysis approaches: orthographic, phonetic, or temporal comparison.

For evaluation, we compared against human-selected transcription pairs for all of our experiments. The benefit of using human annotations is that our reference data was hand-corrected and thus contained little to no “noise.” The drawback, of course, is the resources required to perform the human evaluations: humans who could fluently read each of the languages in the pair (English/Chinese and English/Arabic for our experiments) and the time required to evaluate and select from amongst the possible pairs.

Another question of evaluation concerns intrinsic evaluation versus extrinsic. For our experiments we focused only on extrinsic evaluation, though for future work we would be interested in evaluating the effects of improving transcription pair validation on machine translation and automatic speech synthesis and recognition.

3.5.2.1 Orthographic Match

Orthographic match between two terms in different scripts is obviously a difficult task. Section 3.4 demonstrated the success of using alternative orthography, achieved by mapping each term to a common script, by matching English letters to Chinese Pinyin characters. For an intuitive example, consider matching the sequence “Wangari Maathai” to its corresponding Chinese transcription, “旺加里马塔伊.” This is a much more difficult task than matching the English letters to the sequence of romanized Pinyin characters, ‘ma ta yi,’ which suddenly becomes a much more straightforward task. Mapping the terms into the alternative orthography may require some language-specific knowledge, though there are some existing tools such as the Buckwalter Morphological Analyzer to assist with such tasks.

3.5.2.2 Phonetic Match

Phonetic match is an appealing alternative to orthographic match. In phonetic matching, we map each term in a pair to its pronunciation. We then match the two pronunciation sequences. For example, rather than comparing ‘Maathai’ to ‘马塔伊’, we compare the phonetic sequences ‘m A: t a I’ to ‘m a th a i’. This method can use many of the same algorithms as orthographic match, such as calculating the minimum edit-distance between two strings, but without the problems of using two different scripts. Pronunciation sequences for different languages use the same set of phonetic symbols, modulo the differences in phonetic space of two languages (for example, English has no equivalent for the Arabic sound of ﮑ).

There are a few drawbacks to using phonetic sequences to match two terms. First, phonetic sequences are only an estimate of the correct pronunciation(s) for a term. There may be multiple correct pronunciations for any given term, with little to no way of determining which pronunciation would be most appropriate for a given pair of terms. The estimated pronunciation is typically determined by a speech-to-text system, and may be incorrect; such systems can often be inaccurate particularly on terms such as proper names, which are often the terms under consideration for transcription. Inaccuracies in the pronunciation modeling might lead to inaccuracies in the transcription matching.
Second, matching pairs of phonetic sequences might be difficult due to systematic phonetic substitutions between language-pairs. While the similarity between the two pronunciations can be straightforwardly calculated as a Levenshtein distance, the cost matrix used in the calculation will affect the quality of the phonetically-matched pairs. For example, the cost matrix might be defined as a simple identity matrix, where only identical phones have a substitution cost of zero, and all other costs are constant. Alternatively, a hand-tuned cost matrix could take into account standard phonetic information such as place of articulation and location. Furthermore, such a hand-tuned matrix could incorporate details specific to a given language-pair, such as the phonetic substitutions observed in second-language learners. Creating such a cost matrix requires language-specific knowledge and time spent on creating and tuning the costs.

Figure 3.9 shows the orthographic representation—in English text, foreign script, and mapping the foreign script to an alternative orthography—and phonetic sequences of several transcription pairs.

3.5.2.3 Temporal Distributional Similarity

In temporal distributional similarity matching, we take advantage of the fact that names and terms in news articles tend to be “bursty,” because the coverage of new events causes a sudden increase in the occurrence of related names and terms around the globe. By tracking terms across time in comparable corpora, we can estimate whether two terms tend to occur at during the same period of time.

We follow Tao and Zhai (2005); Tao et al. (2006) in using the Pearson correlation co-efficient to calculate the similarity of two terms’ temporal distributional characteristics. We could also use the log-likelihood ratio or other similar metrics.

There are a few challenges with such an approach. The most obvious is that the “related” terms discovered by this approach may be topically related, such as tsunami and earthquake, but may not be valid transcription pairs. Secondly, one must define “co-occurrence” carefully: clearly two terms that occur in news articles on the same day would be co-occurring terms, but what about within one day of the other? Two days? A week? These decisions can affect the utility of this approach. Finally, the correlation values calculated for each given pair of terms may differ if calculated over a different set of documents, and scaling up to large sets of documents may be challenging.

Figure 3.10 shows the temporal distribution graphs of three sets of transcribed terms as they occurred over the last six months of 2004. In Figure 3.10(a), the English, Chinese, and Arabic terms are all correctly matched. However, the graph for these three terms may appear somewhat noisy; some of the peaks (or “bursts”) are aligned, particularly around Day 100, but other peaks seem to appear independently of the occurrences of the terms in other languages. This example demonstrates that temporal distributions for even the correct pairs are not necessarily perfectly aligned.

The example in Figure 3.10(b) serves to demonstrate the effect of data sparsity on this matching technique. For this example, the English and Chinese terms are correctly paired and have very similar distributional patterns: both terms have a large peak in occurrences just before Day 100 and two large peaks after Day 150. However, the terms do occur relatively infrequently overall; perhaps the correct Arabic term was not contained in the comparable documents, and thus the incorrect Arabic term shown here was provided as the closest match despite the obvious mismatch in temporal distributions.

Similarly, the incorrectly-matched Chinese term shown in Figure 3.10(c) displays a temporal distribution with several extremely high peaks and very few small peaks; nearly the opposite from the temporal distributions of the correctly-matched English and Arabic terms. However, the Chinese term might be rare enough that its co-occurrence with the English term gave the English-Chinese pair a higher ranking than other possible Chinese terms.

Just these few examples have shown some of the mistakes that can be made by using temporal correlation to match transcription pairs.

3.5.2.4 Perceptron Classifier

Section 3.4 described several perceptron classifiers for determining the “validity” of a transcription pair. The described system is a binary classifier, classifying each pair as valid or not, (0,1), and we can use these same systems to classify our dataset. We also can extract the confidence score of the classifier for use in score combination.
Figure 3.10: Examples of the distributional similarity of the occurrences over time of English terms with their transcription pairs in both Chinese and Arabic. *Indicates a correct pairing.

3.5.2.5 Score Combination

Figure 3.11 provides a few examples to motivate our work with score combination. These examples indicate just a few of the cases where the best-matched pair returned by the phonetic-match approach was incorrect but the best-matched pair returned by the temporal-correlation approach was correct, and vice versa.

There are many intuitive reasons to pursue a score combination approach. First, this approach can be used to combine evidence collected from several different sources rather than being forced to select just one knowledge source. Second, previous work has shown the advantages of score combination in classifying valid transcription pairs (Tao et al., 2006). Furthermore, the advantages of score combination—or system combination—have also been demonstrated in other areas, including parsing (Hollingshead et al., 2005) and automatic speech recognition (Fiscus, 1997; Beyerlein, 1998).

There are a few minor drawbacks to using a linear combination approach. First, an unweighted combination is nearly never optimal, so a scaling value $\alpha$ must be optimized against some heldout dataset for each knowledge source used in combination. Second, and a very subtle point, scores are often normalized over an $n$-best list before combination using a soft-max algorithm which assumes a normal distribution of the scores. Since scores rarely follow a normal distribution but tend more towards either binomial or exponential distributions, normalizing might skew the results of the combination. Third, linear combination is a fairly coarse method which does not allow for the possibility that one knowledge source may be preferred under certain conditions and another knowledge source under different conditions. Thus it might be beneficial in future work to explore more fine-grained methods of score combination.
Table 3.11: Examples of the best-matched transcription pairs according to the phonetic-match approach and a temporal correlation-match approach. Only the term marked by an ∗ is a correct match.

<table>
<thead>
<tr>
<th>English</th>
<th>Phonetic-best Match</th>
<th>Correlation-best Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bondevik</td>
<td>班德维克 ∗</td>
<td>成及印度尼西亚</td>
</tr>
<tr>
<td></td>
<td>bang de wei ke</td>
<td>bo ji yin ni xi ya</td>
</tr>
<tr>
<td>Canada</td>
<td>加拿大 ∗</td>
<td>及内地</td>
</tr>
<tr>
<td></td>
<td>jia na da</td>
<td>ji nei di</td>
</tr>
<tr>
<td>Angeles</td>
<td>แอนเจลิส ∗</td>
<td>الدراسة</td>
</tr>
<tr>
<td></td>
<td>'njlys</td>
<td>'ldr'sh</td>
</tr>
<tr>
<td>Lebanese</td>
<td>日本地</td>
<td>黎巴嫩 ∗</td>
</tr>
<tr>
<td></td>
<td>ri ben di</td>
<td>li ba nen</td>
</tr>
<tr>
<td>Gazprom</td>
<td>كيسوفيم</td>
<td>جازبروم ∗</td>
</tr>
<tr>
<td></td>
<td>kyswfm</td>
<td>j'zbrwm</td>
</tr>
</tbody>
</table>

Figure 3.11: Examples of the best-matched transcription pairs according to the phonetic-match approach and a temporal correlation-match approach. Only the term marked by an ∗ is a correct match.

### 3.5.3 Experimental Setup

To conduct our experiments, we extracted “interesting” tokens from the comparable corpora. The corpora used for all experiments herein consisted of the English, Chinese, Arabic Gigaword corpora and news from the Xinhua news agency, augmented with articles from CNA (for Chinese) and AFP (for Arabic).

“Interesting” tokens were defined as those tokens likely to be transcribed, rather than translated, into another language, such as proper names. We filtered the corpora according to the following rules:

- Longer than three characters
- Occurs more than seven times in the entire corpus
- English tokens were initial-capped throughout the entire corpus
- Chinese tokens consisted of only the 1250 characters commonly used for transcription in Chinese

Note that the English and Chinese token extraction is more heavily restricted than the Arabic token extraction, which will result in many more Arabic tokens in the dataset than either of the other two languages. Each extracted English token was paired against every Chinese and Arabic token extracted from news articles published within one day of the news article from which the English token was extracted.

We calculated term relative frequencies for each of the English-Chinese and English-Arabic pairs across the set of news articles published during the last six months of the year 2004. The temporal distribution-match was calculated as the Pearson correlation co-efficient of these relative frequency values. The phonetic-match of each pair was calculated using a hand-crafted cost matrix. We set an arbitrary hard-limit using the phonetic-match score, and removed any pair with a phonetic-match score below the threshold value of ten. If the thresholded set contained more than ten paired terms, then anything ranked below the top ten candidates were also removed. The result is an $n$-best list of at most ten transcription-pair candidates.

### 3.5.4 Results: Ranking

We created a test-set by hand-evaluating the validity of each possible transcription pair extracted from news articles published within one day of 30 December 2004. We report on several evaluation metrics, including accuracy and mean-reciprocal rank (MRR). Accuracy simply indicates how frequently the top-ranked candidate in the $n$-best list is a valid transcription pair. MRR provides “partial-credit” for $n$-best lists that contain a valid pair but that pair is not top-ranked in the list. The higher-ranked the correct candidate is, the more partial-credit will be awarded by the MRR metric. Recall from Section 3.5.2 that there are several reasons why an $n$-best list might not contain a valid transcription pair at all: the English term is not a named entity and therefore should not be transcribed; the correct match for the English term is a mix of translation and transcription, which our algorithms may not find; the correct match is not contained in the corresponding foreign-language document; or the correct match was found but had a low phonetic-match score that pushed the match below the threshold of the list.
Table 3.4: Results on ranking English-Chinese or English-Arabic transcription pairs, mined from comparable corpora. Top three rows are single-score results; bottom four rows are linear-combination results. MRR: Mean reciprocal-rank.

<table>
<thead>
<tr>
<th>Scoring</th>
<th>English ↔ Chinese</th>
<th>English ↔ Arabic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Phonetic</td>
<td>0.0827</td>
<td>0.0742</td>
</tr>
<tr>
<td>Temporal correlation</td>
<td>0.0643</td>
<td>0.0443</td>
</tr>
<tr>
<td>Perceptron (L2P)</td>
<td><strong>0.0923</strong></td>
<td><strong>0.0906</strong></td>
</tr>
<tr>
<td>Phonetic + Correlation</td>
<td>0.0833</td>
<td>0.0751</td>
</tr>
<tr>
<td>Phonetic + Perceptron</td>
<td>0.0909</td>
<td>0.0877</td>
</tr>
<tr>
<td>Correlation + Perceptron</td>
<td>0.0869</td>
<td>0.0809</td>
</tr>
<tr>
<td>Phonetic + Correlation + Perceptron</td>
<td>0.0913</td>
<td>0.0886</td>
</tr>
</tbody>
</table>

The top three rows of Table 3.4 show the results of three approaches to ranking possible transcription pairs: phonetic-match, temporal correlation–match, and discriminatively trained perceptron. The perceptron models used for these experiments were the letter-to-phoneme models trained on data extracted from dictionaries of named-entity pairs, as described in Section 3.4. The English-Chinese perceptron is trained on 60,000 transcription pairs, and outperformed all other ranking methods reported here. In contrast, the English-Arabic perceptron was limited to only 1750 transcription pairs in training, which may have affected its performance on this task.

The bottom four rows of Table 3.4 show the results of several extremely simple methods of score combination. The unweighted linear combination of the phonetic, temporal correlation, and perceptron scores outperformed all other methods for matching English-Arabic terms.

### 3.5.5 Results: Classification

The results reported in the previous section focused on the ranking accuracy of our systems. However, as has been mentioned several times already, it is possible that the “correct” match simply cannot be found within the dataset. In fact, over 80% of the terms in our test set did not have a correct match according to our evaluators. Thus, rather than treat the problem as a ranking task, perhaps it would be more reasonable to treat the problem as a classification task.

We can easily extend the perceptron model to the task of classification rather than ranking by simply setting a threshold on the perceptron scores; the scores range from [0,1], so scores above 0.5 are treated as class 1, and below are treated as class 0. However, we cannot use the phonetic-match or temporal correlation–match models as classifiers; as mentioned previously, the scores of these models are not normalized and thus setting a threshold as we did for the perceptron models is not possible. Therefore, we trained a class-specific discriminative model combination (DMC) model (Beyerlein, 1998) using, as features, the scores output by each of the approaches listed in Table 3.4. The DMC model requires supervised learning, so we were limited to training on the human-evaluated data. Testing on this training data (i.e., a “cheating” experiment), we reduced the error rates on the English-Chinese dataset from 48% to 14%, and from 50% to 20% error on the English-Arabic dataset. We then trained the DMC model using a crossfold, leave-one-out training scenario.

Rather than reporting on one-best accuracy or MRR, in this section we report on the percentages of true-positives and false-negatives from each classifier. Table 3.5 shows the results for the perceptron models and the DMC models trained on English-Chinese and English-Arabic data. Note that the English-Arabic perceptron, which is trained on much less data than the English-Chinese perceptron, achieves a higher true-positive rate at 93%, but the false-negative rate is also higher at 33%.

The perceptron models are optimized for classification accuracy. The DMC models, in contrast, were tuned to maximize true-positives while minimizing false-negatives. The result is a higher rate of false positives, as seen in the table where the English-Chinese DMC model has a false-positive rate of 24% and the English-Arabic DMC model has a false-positive rate of 34%. The rationale behind this choice was that identifying and eliminating invalid transcription pairs was straightforward, if tedious, for our human evaluators, whereas finding valid pairs that were missed during the extraction was not possible. Since the intended purpose of the transcription pairs was for extrinsic evaluation, we tuned the DMC model for high
### Table 3.5: True-positive matrices on classifying English-Chinese or English-Arabic transcription pairs as either valid ("positive") or invalid ("negative") pairs.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>English ↔ Chinese</th>
<th>English ↔ Arabic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>Perceptron</td>
<td>0.83</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>0.92</td>
</tr>
<tr>
<td>DMC</td>
<td>0.76</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>0.11</td>
<td>0.89</td>
</tr>
</tbody>
</table>

recall. If the pairs were intended for direct use in a pipeline system, then we would have tuned the model for high precision.

#### 3.5.6 Discussion & Future Work

The DMC models presented in the previous section have much greater potential than we were able to explore for this work. We trained these models on extremely limited data; increasing the size of the training set would certainly improve performance as well as allowing for more flexibility in model-tuning. The training set could be increased either by including known transcription pairs, such as those extracted from named-entity dictionaries, or by hand-evaluating a larger list of automatically-extracted transcription pairs. Phonetic and temporal distribution data would, of course, have to be calculated for the additional data (which is not an insignificant task). Given a larger training set, we could also train the DMC models as quadratic classifiers, which may provide a larger gain than the linear classifiers that we examined here.

The DMC models also present an opportunity to create a more fine-grained linear combination model. If we were to divide the data into finer-grained classes, the DMC model could train separate weights for each class. Furthermore, by performing an error analysis on the output of the existing DMC models, we might be better able to understand some of the conditions under which the different scoring methods perform better than the others, thus providing a starting point for defining finer-grained classes of possible transcription pairs.

There were several limitations to the experiments presented here. The \( n \)-best lists of transcription pairs were thresholded by phonetic-match score, then scored for temporal correlation. This ordering with a hard threshold may have removed from consideration some pairs with high temporal correlation, limiting the possibilities for success using the score combination approach. The \( n \)-best lists were also severely restricted in size in order to allow for human evaluation; with access to more evaluators or allowing more time for the evaluation, we could have increased the size of the lists and, perhaps, increased the number of valid pairs included in the dataset.

Much of the data obtained from comparable corpora was noisy. We might be able to obtain better matched-pairs from web-mined data (see Section 2; the pronunciation data found from web-mining might also provide a more accurate source of pronunciations than speech synthesis systems.

The transcription pairs mined, analyzed, and filtered by the experiments described here could have several uses. The paired data could be used to estimate cross-lingual phonetic substitution cost matrices, to provide predicted pronunciations of non-native speakers, or to “anchor” the output of a machine translation system either at the decoding level as described by Hermajakob et al. (2008) or earlier in the machine translation pipeline such as during word-alignment or phrasal-extraction.

#### 3.6 ScriptTranscriber

One of the goals of this workshop was to develop a set of public-domain tools for doing transcription. Specifically, the goal was to provide modules for producing guesses at pronunciations for any word in any script, for computing edit distances between strings using a variety of measures, for computing time correlations, and providing a set of prepackaged recipes for mining possible transcription pairs from comparable corpora.

The toolkit ScriptTranscriber consists of approximately 7,500 lines of object-oriented Python. Some of the modules require PySNoW, the Python interface to the SNoW machine-learning package Carlson et al.
The modules of ScriptTranscriber are as follows:

1. **XML document structure module**, an example of which is shown in Figure 3.12. The top-level XML representation consists of a set of tupled documents, ordered according to some reasonable criterion such as time. Each doc elements consists of one or more lang elements, which represent the original document(s) in the named language. Within each doc, the lang documents are intended to be comparable documents. For example, in Figure 3.12 the English document containing Clinton is assumed to be parallel to the Chinese one containing 克林顿. Within each doc are a set of tokens, in no particular order, which represent terms—typically names—that have been extracted during the term extraction phase described below, along with a set of possible pronunciations and their counts.

2. **Term extractor class** to extract interesting terms from raw text, i.e. terms that are likely to be transcribed across scripts. We provide four specializations of this:
   - A simple capitalization-based extractor that looks for sentence medial capitalized terms if the script supports capitalization; otherwise just returns all terms.
   - Chinese foreign name extractor. This extractor uses a list of characters that are commonly used to transcribe foreign words in Chinese, and extracts sequences of at least three such characters.
   - Chinese personal name extractor. This uses a list of family names to find possible Chinese personal names.
   - Thai extractor. This uses a discriminative model (built using SNoW) to predict word boundaries in unsegmented Thai text, and then returns all found terms.

Users can easily define their own extractors so that, for example, if they have a good named entity extractor for a language, they can simply define an interface to that as a derived class of Extractor.

3. **A morphological analyzer class**, a placeholder for a range of possible morphological analyzers. The one provided looks for words that share common substrings and groups them into tentative morphological equivalence classes, along the lines of Klementiev and Roth (2006).

4. **Pronouncer module**. There are three specializations provided:
   - Unitran, which provides guesses on pronunciations for most grapheme code points in the Basic Multilingual Plane that are also used as scripts for languages. (For example, IPA is not covered.)

---

1. [http://l2r.cs.uiuc.edu/~cogcomp/](http://l2r.cs.uiuc.edu/~cogcomp/) note that the PySNoW package, is not provided with ScriptTranscriber: that must be downloaded separately from the CCG website. Users should follow the directions for compiling snow.o, required by ScriptTranscriber.
- English pronouncer: provides a Festival-derived pronunciations Taylor et al. (1998) for about 2.9 million words.
- Hanzi (Chinese character) pronouncer. Provides Mandarin and Native Japanese (kunyomi) pronunciations for characters. Only one pronunciation prediction is provided for each character.

5. Comparator module, which provides the cost for the mapping between strings. Three specializations are provided:

- Hand-built phonetic comparator, which uses the phonetic distance method of Tao et al. (2006); Yoon et al. (2007)
- Perceptron-based comparator, described in Section 3.4.
- Time correlation comparator, described in Section 3.5.

A sample use of the program is given on the next page. This program loads some Thai and English data from the distributed testdata directory, extract terms from each, builds and dumps an XML document representation, and computes phonetic distances for each pair of terms in each document.
#!/bin/env python
# -*- coding: utf-8 -*-

"""Sample transcription extractor based on the LCTL Thai parallel
data. Also tests Thai prons and alignment.
"""

__author__ = "rws@uiuc.edu (Richard Sproat)"

import sys
import os
import documents
import tokens
import token_comp
import extractor
import thai_extractor
import pronouncer
from __init__ import BASE_

## A sample of 10,000 from each:

ENGLISH_ = '%s/testdata/thai_test_eng.txt' % BASE_
THAI_ = '%s/testdata/thai_test_thai.txt' % BASE_
XML_FILE_ = '%s/testdata/thai_test.xml' % BASE_
MATCH_FILE_ = '%s/testdata/thai_test.matches' % BASE_
BAD_COST_ = 6.0

def LoadData():
    t_extr = thai_extractor.ThaiExtractor()
    e_extr = extractor.NameExtractor()
    doclist = documents.Doclist()
    doc = documents.Doc()
    doclist.AddDoc(doc)
    #### Thai
    lang = tokens.Lang()
    lang.SetId('th')
    doc.AddLang(lang)
    t_extr.FileExtract(THAI_)
    lang.SetTokens(t_extr.Tokens())
    lang.CompactTokens()
    for t in lang.Tokens():
        pronouncer_ = pronouncer.UnitranPronouncer(t)
        pronouncer_.Pronounce()
    #### English
    lang = tokens.Lang()
    lang.SetId('en')
    doc.AddLang(lang)
    e_extr.FileExtract(ENGLISH_)
    lang.SetTokens(e_extr.Tokens())
    lang.CompactTokens()
    for t in lang.Tokens():
pronouncer_ = pronouncer.EnglishPronouncer(t)
pronouncer_.Pronounce()
return doclist

def ComputePhoneMatches(doclist):
    matches = {}
    for doc in doclist.Docs():
        lang1 = doc.Langs()[0]
        lang2 = doc.Langs()[1]
        for t1 in lang1.Tokens():
            hash1 = t1.EncodeForHash()
            for t2 in lang2.Tokens():
                hash2 = t2.EncodeForHash()
                try: result = matches[(hash1, hash2)] ## don't re-calc
                except KeyError:
                    comparator = token_comp.OldPhoneticDistanceComparator(t1, t2)
                    comparator.ComputeDistance()
                    result = comparator.ComparisonResult()
                    matches[(hash1, hash2)] = result
    values = matches.values()
    values.sort(lambda x, y: cmp(x.Cost(), y.Cost()))
    p = open(MATCH_FILE_, 'w') ## zero out the file
    p.close()
    for v in values:
        if v.Cost() > BAD_COST_: break
        v.Print(MATCH_FILE_, 'a')

if __name__ == '__main__':
    doclist = LoadData()
    doclist.XmlDump(XML_FILE_, utf8 = True)
    ComputePhoneMatches(doclist)
3.7 Ongoing Work

In addition to the work reported here, we also did some preliminary work on other approaches for distance comparisons. Following Kuo et al. (2007), we tried using ASR confusions between English and Mandarin to produce phonetic distance scores. In collaboration with the Speech Recognition subteam, we ran the IBM Mandarin recognizer on 2,689 utterances of English RT04, and extracted confusions between phones from the one-best output of the recognizer. We then used this distance measure with our Chinese-English name dictionary, where the task was, for each English word, find the correct Chinese word among a set of five words, the other four of which were randomly selected foils. Unfortunately this method did not work even as well as the hand-built phonetic scores from Tao et al. (2006); Yoon et al. (2007), as the following Mean Reciprocal Rank and Accuracy scores show:

<table>
<thead>
<tr>
<th></th>
<th>MRR</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.86</td>
<td>0.76</td>
</tr>
<tr>
<td>ASR</td>
<td>0.82</td>
<td>0.69</td>
</tr>
</tbody>
</table>

However, we believe that this approach could eventually turn out to be useful, if we can derive more reasonable confusion matrices from the ASR system. Of course, the downside of such an approach is that it is not readily applicable to arbitrary language pairs: one does not always have the resources in terms of a large vocabulary ASR system for one language and a large transcribed speech corpus for the other.

At the time of writing we are also continuing to investigate methods for combining string distance and temporal cooccurrence scores, and we hope to be able to report on this in future work.
Chapter 4

Multilingual Spoken Term Detection

The first part of the report from the speech subteam focusses on the effect of pronunciations for Out-of-Vocabulary (OOV) query terms on the performance of a spoken term detection (STD) task. OOV terms, typically proper names or foreign language terms occur infrequently but are rich in information. The STD task returns relevant segments of speech that contain one or more of these OOV query terms. The STD system described here indexes word-level and subword level lattices produced by an LVCSR system using Weighted Finite State Transducers (WFST). Experiments comparing pronunciations using n-best variations from letter-to-sound rules, morphing pronunciations using phone confusions for the OOV terms and indexing one-best transcripts, lattices and confusion networks are presented. The following observations are worth mentioning: phone indexes generated from subwords represented OOVs well, too many variants for the OOV terms degrades performance if pronunciations are not weighted. Finally, we also studied the use of web-based pronunciations.

The second part of the report addresses selecting between candidate pronunciations for out-of-vocabulary words in speech processing tasks. We introduce a simple, unsupervised method that outperforms the conventional supervised method of forced alignment with a reference. The success of this method is independently demonstrated using three metrics from large-scale speech tasks: word error rates for large vocabulary continuous speech recognition, decision error tradeoff curves for spoken term detection, and phone error rates compared to a handcrafted pronunciation lexicon. The experiments were conducted using state-of-the-art recognition, indexing, and retrieval systems. The results were compared across many terms, hundreds of hours of speech, and well known data sets.

4.1 Part I - Effect of Pronunciations on OOV Queries in Spoken Term Detection

4.1.1 Introduction

The rapidly increasing amount of spoken data calls for solutions to index and search this data. Spoken term detection (STD) is a key information retrieval technology which aims open vocabulary search over large collections of spoken documents. The major challenge faced by STD is the lack of reliable transcriptions, an issue that becomes even more pronounced with heterogeneous, multilingual archives. Considering the fact that most STD queries consist of rare named entities or foreign words, retrieval performance is highly dependent on the recognition errors. In this context, lattice indexing provides a means of reducing the effect of recognition errors by incorporating alternative transcriptions in a probabilistic framework.

The classical approach consists of converting the speech to word transcripts using large vocabulary continuous speech recognition (LVCSR) tools and extending classical Information Retrieval (IR) techniques to word transcripts. However, a significant drawback of such an approach is that search on queries containing out-of-vocabulary (OOV) terms will not return any result. These words are replaced in the output transcript by alternatives that are probable, given the acoustic and language models of the ASR. It has been experimentally observed that over 10% of user queries can contain OOV terms Logan et al. (2002), as queries often
relate to named entities that typically have a poor coverage in the ASR vocabulary. The effects of OOV query terms in spoken data retrieval are discussed in Woodland et al. (2000). In many applications, the OOV rate may get worse over time unless the recognizer’s vocabulary is periodically updated. An approach for solving the OOV issue consists of converting the speech to phonetic transcripts and representing the query as a sequence of phones. Such transcripts can be generated by expanding the word transcripts into phones using the pronunciation dictionary of the ASR system. Another way would be to use subword (phones, syllables, or word-fragments) based language models. The retrieval is based on searching the sequence of subwords representing the query in the subword transcripts. Some of these works were done in the framework of the NIST TREC Spoken Document Retrieval tracks in the 1990s and are described by Garofolo et al. (2000). Popular approaches are based on search on subword decoding Clements et al. (2002); Seide et al. (2004); Saraclar and Sproat (2004); Siohan and Bacchiani (2005); Mamou et al. (2007a) or search on the subword representation of word decoding enhanced with phone confusion probabilities and approximate similarity measures for word Chaudhari and Picheny (2007).

Other research works have tackled the OOV issue by using the IR technique of query expansion. In classical text IR, query expansion is based on expanding the query by adding additional words using techniques like relevance feedback, finding synonyms of query terms, finding all of the various morphological forms of the query terms and fixing spelling errors. Phonetic query expansion has been used by [Li00] for Chinese spoken document retrieval on syllable-based transcripts using syllable-syllable confusions from the ASR.

The rest of this part of the report is organized as follows. In Section 4.1.2 we explain the methods used for spoken term detection. These include the indexing and search framework based on WFSTs, formation of phonetic queries using letter to sound models, and expansion of queries to reflect phonetic confusions. In Section 4.1.3 we describe our experimental setup and present the results. Finally, in Section 4.1.6 we summarize our contributions.

### 4.1.2 Methods

#### 4.1.2.1 WFST-based Spoken Term Detection

General indexation of weighted automata provides an efficient means of indexing speech utterances based on the within utterance expected counts of substrings (factors) seen in the data Allauzen et al. (2004); Saraclar and Sproat (2004). In the most basic form, mentioned algorithm leads to an index represented as a weighted finite state transducer (WFST) where each substring (factor) leads to a successful path over the input labels for each utterance that particular substring was observed. Output labels of these paths carry the utterance ids, while path weights give the within utterance expected counts. The index is optimized by weighted transducer determinization and minimization Mohri et al. (1996) so that the search complexity is linear in the sum of the query length and the number of indices the query appears. Figure 4.1.a illustrates the utterance index structure in the case of single-best transcriptions for a simple database consisting of two strings: “a a” and “b a”.

![Diagram of Utterance Index](image1)

![Diagram of Modified Utterance Index](image2)

Figure 4.1: Index structures

Explained construction is ideal for the task of utterance retrieval where the expected count of a query term within a particular utterance is of primary importance. In the case of STD, this construction is still useful as the first step of a two stage retrieval mechanism Parlak and Saraclar (2008a) where the retrieved utterances are further searched or aligned to determine the exact locations of queries since the index provides the utterance information only. One complication of this setup is that each time a query term occurs within
an utterance, it will contribute to the expected count within that particular utterance and the contribution of distinct instances will be lost. Here we should clarify what we refer to by an occurrence and an instance. In the context of lattices where arcs carry recognition unit labels, an occurrence corresponds to any path comprising of the query labels, an instance corresponds to all such paths with overlapping time-alignments. Since the index provides neither the individual contribution of each instance to the expected count nor the number of instances, both of these parameters have to be estimated in the second stage which in turn compromises the overall detection performance.

To overcome some of the drawbacks of the two-pass retrieval strategy, a modified utterance index which carries the time-alignment information of substrings in the output labels was created. Figure 4.1.b illustrates the modified utterance index structure derived from the time-aligned version of the same simple database: “a₀→₁ a₁→₂” and “b₀→₁ a₁→₂”. In the new scheme, preprocessing of the time alignment information is crucial since every distinct alignment will lead to another index entry which means substrings with slightly off time-alignments will be separately indexed. Note that this is a concern only if we are indexing lattices, consensus networks or single-best transcriptions do not have such a problem by construction. Also note that no preprocessing was required for the utterance index, even in the case of lattices, since all occurrences in an utterance were identical from the indexing point of view (they were in the same utterance). To alleviate the time-alignment issue, the new setup clusters the occurrences of a substring within an utterance into distinct instances prior to indexing. Desired behavior is achieved via assigning the same time-alignment information to all occurrences of an instance.

Main advantage of the modified index is that it distributes the total expected count among instances, thus the hits can now be ranked based on their posterior probability scores. To be more precise, assume we have a path in the modified index with a particular substring on the input labels. Weight of this path corresponds to the posterior probability of that substring given the lattice and the time interval indicated by the path output labels. The modified utterance index provides posterior probabilities compared to expected counts provided by the utterance index. Furthermore, second stage of the previous setup is no longer required since the new index already provides all the information we need for an actual hit: the utterance id, begin time and duration. Eliminating second stage significantly improves the search time since time-alignment of utterances takes much more time compared to retrieving them. On the other hand, embedding time-alignment information leads to a much larger index since common paths among different utterances are largely reduced by the mismatch between time-alignments which in turn compromises the effectiveness of the weighted automata optimization. To smooth this effect out, time-alignments are quantized to a certain extent during preprocessing without altering the final performance of the STD system.

Searching for a user query is a simple weighted transducer composition operation Mohri et al. (1996) where the query is represented as a finite state acceptor and composed with the index from the input side. The query automaton may include multiple paths allowing for a more general search, i.e. searching for different pronunciations of a query word. The WFST obtained after composition is projected to its output labels and ranked by the shortest path algorithm to produce results Mohri et al. (1996). In effect, we obtain results with decreasing posterior scores.

Figure 4.2 compares the proposed system with the 2-pass retrieval system on the stddev06 data-set in a setup where dryrun06 query-set, word-level ASR lattices and word-level indexes are utilized. As far as Detection Error Tradeoff (DET) curves are concerned, there is no significant difference between the two methods. However, proposed method has a much shorter search time, a natural result of eliminating time-costly second pass.

4.1.2.2 Query Forming and Expansion for Phonetic Search

When using a phonetic index, the textual representation of a query needs to be converted into a phone sequence or more generally a WFST representing the pronunciation of the query. For OOV queries, this conversion is achieved using a letter-to-sound (L2S) system. In this study, we use n-gram models over (letter, phone) pairs as the L2S system, where the pairs are obtained after an alignment step. Instead of simply taking the most likely output of the L2S system, we investigate using multiple pronunciations for each query. Assume we are searching for a letter string $l$ with the corresponding phone-strings set $\Pi_n(l)$ : n-best L2S pronunciations. Then the posterior probability of finding $l$ in lattice $L$ within time
Figure 4.2: Comparison of 1-pass & 2-pass strategies in terms of retrieval performance and runtime.

The interval $T$ can be written as

$$P(l|L, T) = \sum_{p \in \Pi_n(l)} \hat{P}(l|p)P(p|L, T)$$

where $P(p|L, T)$ is the posterior score supplied by the modified utterance index and $\hat{P}(l|p)$ is the posterior probability derived from L2S scores.

Composing an OOV query term with the L2S model returns a huge number of pronunciations of which unlikely ones are removed prior to search to prevent them from boosting the false alarm rates. To obtain the conditional probabilities $\hat{P}(l|p)$, we perform a normalization operation on the retained pronunciations which can be expressed as

$$\hat{P}(l|p) = \frac{P(\alpha(l, p)|l)}{\sum_{\pi \in \Pi_n(l)} P(\alpha(l, \pi)|l)}$$

where $P(l, p)$ is the joint score supplied by the L2S model and $\alpha$ is a scaling parameter. Most of the time, retained pronunciations are such that a few dominate the rest in terms of likelihood scores, a situation which becomes even more pronounced as the query length increases. Thus, selecting $\alpha = 1$ to use raw L2S scores leads to problems since most of the time best pronunciation takes almost all of the posterior probability leaving the rest out of the picture. The quick and dirty solution is to remove pronunciation scores instead of scaling them. This corresponds to selecting $\alpha = 0$ which assigns the same posterior probability $\hat{P}(l|p)$ to all pronunciations: $\hat{P}(l|p) = 1/|\Pi_n(l)|$, for each $p \in \Pi_n(l)$. Although simple, this method is likely to boost false alarm rates since it does not make any distinction among pronunciations. The challenge is to find a good query-adaptive scaling parameter which will dampen the large scale difference among L2S scores. In our experiments we selected $\alpha = 1/|l|$ which scales the log likelihood scores by dividing them with the “length of the letter string”. This way, pronunciations for longer queries are effected more than those for shorter ones. Another possibility is to select $\alpha = 1/|p|$, which does the same with the “length of the phone string”. Section 4.1.3.2 presents a comparison between removing pronunciation scores and scaling them with our method.

Similar to obtaining multiple pronunciations from the L2S system, the queries can be extended to similar sounding ones by taking phone confusion statistics into account. In this approach, the output of the L2S system is mapped to confusable phone sequences using a sound-to-sound (S2S) system. The S2S system is identical to the L2S system except for the input alphabet.
4.1.3 Experiments

4.1.3.1 Experimental Setup

Our goal was to address pronunciation validation using speech for OOVs in a variety of applications (recognition, retrieval, synthesis) for a variety of types of OOVs (names, places, rare/foreign words). To this end we selected speech from English broadcast news (BN) and 1290 OOVs. The OOVs were selected with a minimum of 5 of acoustic instances per word, and common English words were filtered out to obtain meaningful OOVs (e.g. NATALIE, PUTIN, QAEDA, HOLLOWAY), excluding short (less than 4 phones) queries. Once selected, these were removed from the recognizer’s vocabulary and all speech utterances containing these words were removed from training.

The LVCSR system was built using the IBM Speech Recognition Toolkit Soltau et al. (2005) with acoustic models trained on 300 hours of HUB4 data with utterances containing OOV words excluded. The excluded utterances (around 100 hours) were used as the test set for WER and STD experiments. The language model for the LVCSR system was trained on 400M words from various text sources. The LVCSR system’s WER on a standard BN test set RT04 was 19.4%. This system was also used for lattice generation for indexing for OOV queries in the STD task.

4.1.3.2 Results

The baseline experiments were conducted using the reference pronunciations for the query terms, which we refer to as reflex. The L2S system was trained using the reference pronunciations of the words in the vocabulary of the LVCSR system. This system was then used to generate multiple pronunciations for the OOV query words. Further variations on the query term pronunciations were obtained by applying a phone confusion S2S transducer to the L2S pronunciations.

Baseline - Reflex

For the baseline experiments, we used the reference pronunciations to search for the queries in various indexes. The indexes were obtained from word and subword (fragment) based LVCSR systems. The output of the LVCSR systems were in the form of 1-best transcripts, consensus networks, and lattices. The results are presented in Table 4.1. Best performance is obtained using subword lattices converted into a phonetic index.

<table>
<thead>
<tr>
<th>Data</th>
<th>P(FA)</th>
<th>P(Miss)</th>
<th>ATWV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1-best</td>
<td>.00001</td>
<td>.770</td>
<td>.215</td>
</tr>
<tr>
<td>Word Consensus Nets</td>
<td>.00002</td>
<td>.687</td>
<td>.294</td>
</tr>
<tr>
<td>Word Lattices</td>
<td>.00002</td>
<td>.657</td>
<td>.322</td>
</tr>
<tr>
<td>Fragment 1-best</td>
<td>.00001</td>
<td>.680</td>
<td>.306</td>
</tr>
<tr>
<td>Fragment Consensus Nets</td>
<td>.00003</td>
<td>.584</td>
<td>.390</td>
</tr>
<tr>
<td>Fragment Lattices</td>
<td>.00003</td>
<td>.485</td>
<td>.484</td>
</tr>
</tbody>
</table>

L2S

For the L2S experiments, we investigated varying the number of pronunciations for each query for two scenarios and different indexes. The first scenario considered each pronunciation equally likely (unweighted queries) whereas the second made use of the L2S probabilities properly normalized (weighted queries). The results are presented in Figure 4.3 and summarized in Table 4.2. For the unweighted case the performance peaks at 3 pronunciations per query. Using weighted queries improves the performance over the unweighted case. Furthermore, adding more pronunciations does not degrade the performance. Best results are comparable to the reflex results.

39
The DET plot for weighted L2S pronunciations using indexes obtained from fragment lattices is presented in Figure 4.4. The single dots indicate MTWV (using a single global threshold) and ATWV (using term specific thresholds Miller et al. (2007)) points.
For the S2S experiments, we investigated expanding the 1-best output of the L2S system. In order to mimic common usage we used indexes obtained from 1-best word and subword hypotheses converted to phonetic transcripts. As shown in Table 4.3 a slight improvement was obtained when using a trigram S2S system representing the phonetic confusions. These results were obtained using unweighted queries and using weighted queries may improve the results.

<table>
<thead>
<tr>
<th>Lattices</th>
<th># Best</th>
<th>P(FA)</th>
<th>P(Miss)</th>
<th>ATWV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>1</td>
<td>.00002</td>
<td>.795</td>
<td>.190</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>.00002</td>
<td>.785</td>
<td>.192</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>.00003</td>
<td>.778</td>
<td>.193</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>.00004</td>
<td>.775</td>
<td>.189</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>.00004</td>
<td>.771</td>
<td>.185</td>
</tr>
<tr>
<td>Fragments</td>
<td>1</td>
<td>.00002</td>
<td>.757</td>
<td>.228</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>.00002</td>
<td>.748</td>
<td>.230</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>.00003</td>
<td>.742</td>
<td>.229</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>.00004</td>
<td>.738</td>
<td>.227</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>.00004</td>
<td>.736</td>
<td>.221</td>
</tr>
</tbody>
</table>

### 4.1.4 Pronunciations derived from the web

We also explored the use of pronunciations derived from the web provided to us by the rest of the team (details available in this report). For query terms that do not have pronunciations from any web extraction method, we backed-off to the ones generated by a letter-to-sound system. Some of the pronunciations obtained from the web helped fix detection errors, particularly ones corresponding to named entities. The various pronunciations that we explored are:

- Pronunciations obtained from a letter-to-sound system trained on the gold standard (reflex) and n-best variations (L2S 00)
- Ad-hoc pronunciations from the web (Webpron 03)
- Cleaned and normalized IPA pronunciations obtained by mining the web (Webpron 04, 05)
- Hand-cleaned adhoc pronunciations (Webpron 06)
- Best Pronunciation from LVCSR (will be presented with the LVCSR section)
- Union of pronunciations (can have multiple pronunciations for any word from different methods)

Table 4.4 illustrates the number of OOV queries that could be represented by pronunciations from the web after various filtering and cleaning processes.

Figure 4.5 plots the DET curves for different pair wise union of pronunciations from this set. In general, the use of web-based pronunciations, seems to improve the overall ATWV measure.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of OOVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adhoc</td>
<td>343</td>
</tr>
<tr>
<td>Cleaned and Normalized IPA</td>
<td>269</td>
</tr>
<tr>
<td>Hand Cleaned Adhoc</td>
<td>109</td>
</tr>
</tbody>
</table>

Table 4.4: Number of OOVs that had pronunciations on the web
Table 4.5: English results: Analysis of L2S v/s Web-based pronunciations with positive impact

Table 4.6: English results: Analysis of L2S v/s Web-based pronunciations with negative impact

4.1.5 Use of a foreign language: Turkish

A Turkish LVCSR system similar to the English system was built on 100 hours of Turkish broadcast news. Out of the English OOV queries, 445 of them were selected to search through 25 hours of Turkish data. These satisfied the conditions of minimum number of occurrences in the audio to be searched through. Pronunciations for these English words in Turkish data were derived from their phonetic transcriptions. Initial results are presented in Figure 4.7 and seem to validate our indexing and retrieval approach presented here.

Table 4.5: English results: Analysis of L2S v/s Web-based pronunciations with positive impact

Table 4.6: English results: Analysis of L2S v/s Web-based pronunciations with negative impact
4.1.6 Part I - Summary

Phone indexes generated from subwords represent OOVs better than phone indexes generated from words. Modeling phonetic confusions yields slight improvements. Using multiple pronunciations obtained from L2S system improves the performance, particularly when the alternatives are properly weighted. In general, the use of web-based pronunciations showed promising results for certain types of OOV queries. We have also demonstrated the effect of pronunciations when searching for English words through audio in a foreign language.

4.2 Part II - Unsupervised pronunciation validation

4.2.1 Introduction

Several speech processing applications including large vocabulary continuous speech recognition (LVCSR), spoken term detection (STD), and speech synthesis rely on a fixed vocabulary and a pronunciation for each word therein. This pronunciation lexicon typically contains mappings from an orthographic form of a words (e.g. QUEDA) into a phonetic form (e.g. /k aa d ax/) that can be used for decoding, indexing,
retrieval, or synthesis. Although the pronunciation lexicon remains fixed, realistic use requires a constantly
change vocabulary resulting in words that are out-of-vocabulary (OOV). OOVs can be new words, rare
words, foreign words, or words unknown to be important at the time the lexicon was formed. Adjusting to
change in vocabulary demands the generation of pronunciations and a need to automatically select between
candidates. Therefore, this work addresses the question: given a word in any language, and a set of candidate
pronunciations, how can you determine the best pronunciation of that word?

Challenges within OOV modeling and pronunciation validation are not new, but issues with OOV words
have traditionally been given less attention due their low impact on word error rate (WER). Recent work
White et al. (2008); et al. (2008) and the development of the STD task Mamou et al. (2007b) have highlighted
their importance as rare words, which are therefore information rich.

An initial approach to create pronunciations for OOV employs a trained linguist, but they are expensive,
often produce inconsistent representations, generate few pronunciations per hour, and have limited areas of
expertise et al. (1999). Therefore effort has been made toward data-driven pronunciation modeling. Previous
work et al. (1999); Lucassen and Mercer (1984) addressing pronunciation validation comes from pronunciation
modeling attempting to simultaneously generate/validate pronunciations using existing lexica et al. (1999),
linguistic rules et al. (2006), speech samples Beaufays et al. (2003); Ramabhadran et al. (1998), or all of the
above (see literature pronunciation modeling, grapheme-to-phoneme, letter-to-sound). Such work generally
includes a criteria for creating a pronunciation for an OOV that involves the modality of data used to
create it (e.g. generating pronunciations from lexica tests against comparisons to held out entries in the
lexica, generating pronunciations from speech forced-alignment uses accuracy or WER of speech samples).
The previous work on data-driven pronunciation modeling addresses pronunciation variation et al. (1999);
(2003) they concentrates on names and places, directory services, noting that proper names can be hard
where it is difficult to reuse letter-to-sound rules from common words.

For example, in Beaufays et al. (2003) they learn pronunciations from audio samples along with rules from
an existing lexicon and develop an iterative algorithm for pronunciation refinement; accuracy of recognition
on directory assistance samples is measured. For many cases using speech samples including et al. (2006),
the standard score comes from aligning the speech sample of a word against the putative pronunciation,
sometimes with a filler model for likelihood ratio threshold. In et al. (2006) they augment acoustic likelihood
with linguistic features and use a decision tree classifier rather than a threshold; they attempt to verify
pronunciations for literacy assessment and treat the problem as estimating a confidence score over a short
utterance (the word of interest).

This work departs from the standard framework of simultaneously generating and testing pronunciations.
We are agnostic about where candidates come from, isolating the task to choose between them. Furthermore,
we concentrate on a large number of difficult words, of which many are foreign proper names and places.
Our evaluation involves large-scale speech tasks with large data sets in an effort to present results that
generalize. We use two methods to select between candidate pronunciations: a conventional supervised
method via forced-alignment, an unsupervised method via recognition.

We compare these two methods via three metrics: phone error rate (PER) against a reference pronunciation
to analyze the difference with a handcrafted lexicon, WER for LVCSR to see impact on their recognition as
well as their impact on recognizing other words in the vocabulary, and decision error tradeoff (DET) curves for
STD for searching OOVs. The end goal was to identify a methodology for picking correct pronunciations.
This work was conducted as part of the Johns Hopkins University summer workshop (JHUWS08) team
'Multilingual Spoken Term Detection' where pronunciations were generated via letter-to-sound models, those
augmented from web data CITE, or from transliteration models.

4.2.1.1 Baseline Supervised Method

Our baseline mechanism for choosing between two candidate pronunciations was to pick the pronunciation
with higher average acoustic likelihood from a forced-alignment with a reference, with the average taken over
several speech samples.

Performance is measured from approximately 500 words via three metrics: edit distance to a reference
lexicon, WER on decoding 100 hrs of speech, and STD DET curves on the LVCSR lattices for the same 100
hours.
4.2.1.2 Data set, OOV terms, systems

Our goal was to address pronunciation validation using speech for OOVs in a variety of applications (recognition, retrieval, synthesis) for a variety of types of OOVs (names, places, rare/foreign words). To this end we selected speech from English broadcast news (BN) and approximately 500 OOVs. The OOVs were selected with a minimum of 5 of acoustic instances per word, and common English words were filtered out to obtain meaningful OOVs (e.g. NATALIE, PUTIN, QAEDA, HOLLOWAY). Once selected, these were removed from the recognizer's vocabulary and all speech utterances containing these words were removed from training. For each OOV, two candidate pronunciations are considered, each from a variant of a letter-to-sound system. These OOVs were taken from a larger set used to compare web-data augmented letter-to-sound systems, a subset on which two particular letter-to-sounds systems differed. For details the reader is referred to CITE-Bhuv.

The LVCSR system was built using the IBM Speech Recognition Toolkit Soltan et al. (2005) with acoustic models trained on 300 hours of HUB4 data with utterances containing OOV words excluded. The excluded utterances (around 100 hours) were used as the test set for WER and STD experiments. The language model for the LVCSR system was trained on 400M words from various text sources. The LVCSR system's WER on a standard BN test set RT04 was 19.4%. This system was also used for lattice generation for indexing for OOV queries in the STD task along with the OpenFST based Spoken Term Detection system from Bogazici University Parlak and Saraclar (2008b).

4.2.1.3 Supervised validation

Let \( X \) denote a sequence of acoustic observation vectors; the objective of the recognizer is to find the most likely word sequence \( W^* \) given the acoustic vectors:

\[
W^* = \arg \max_W p(W|X) \tag{4.1}
\]

\[
= \arg \max_W p(X|W)p(W) \tag{4.2}
\]

where Equation 2 comes from rewriting Equation 1 using Bayes rule and considering that \( p(X) \) does not play a role in the maximization; \( p(X|W) \) denotes the acoustic likelihood of the acoustic observations given a word sequence hypothesis \( W \); \( p(W) \) is the prior probability of that word sequence \( W \) as defined by a language model.

The conventional method for selecting between pronunciation candidates involves using a transcript and performing a forced alignment against it: during alignment there is a constraint in decoding path \( W \) to the reference transcript (with each word replaced by its pronunciation in the lexicon), augmented with candidate pronunciations. Speech data that contain the OOV are aligned with the acoustic models corresponding to each candidate pronunciation via Viterbi search, and the maximum likelihood acoustic score determines the 'winner' candidate Lucassen and Mercer (1984); et al. (1999); Beaufays et al. (2003); et al. (2006).

Some of the work referenced above attempts improve the decision function or include additional information while simultaneously generating and validating pronunciations. Our work assumes that pronunciations have been provided and seeks to decide between them. Also, this work concentrates on simple and fast methods for large scale heterogeneous applications.

4.2.2 Unsupervised Method

Using standard automatic methods (e.g. Section 4.2.1.3) for verifying pronunciations require transcribed audio, which can cost as much as 100$/hr (common) - 400$/hr (new language) to transcribe. Transcription is time-consuming, laborious, and difficult to recruit/keep labelers for transcribing. However, in many applications meta-data can alleviate the need by pointing to speech likely to contain a word of interest, which can be used to select between candidate pronunciations for that word. For example, items in the news, television shows, etc. are a rich source of untranscribed speech for unsupervised validation.

Moreover, often we do not have access to a transcript corresponding to audio examples of an OOV, but we may have some knowledge it has occurred in an audio archive. For example, we may know from meta-data that a broadcast news episode recently aired about a conflict in Iraq, and at present it would give us
4.2.3 Results

For each of the metrics below, a pronunciation lexicon was created for the set of OOVs (approximately 500). For every OOV there were two candidate pronunciations from different letter-to-sound systems, and we compare the two methods described above for choosing between the two candidates for this set (along with an 'upper-bound' and 'lower-bound'). These 500 words were removed from a handcrafted lexicon, therefore we have a set of 'true' pronunciations. The 'upper'- and 'lower-bound' take advantage of this knowledge, denoted plex – best and plex – worst. plex – best selects the candidate that is the closest (in edit distance) to a reference pronunciation that word, and plex – worst selects the farthest.

For example, in Table 1 two OOVs are listed, each with two hypothesized pronunciations. Here, plex – best would have as members '/k aa d ax/' and '/sh ax v ow/'.

The two methods compared are those described above, where sup – force denotes the lexicon created from selecting pronunciations based on supervised forced-alignment with a reference, and unsup – reco denotes the lexicon created from selection based on unsupervised decoding. For the unsupervised case approximately the time for one broadcast news show was decoded using each candidate pronunciation, making sure to include all the speech examples used for the forced-alignment somewhere in the data.

4.2.3.1 Large vocabulary continuous speech recognition

In addition to comparing methods using the performance in speech tasks, we can see which method produces pronunciations that are closest to a reference. For example in Table 1, if speech had selected the bold pronunciations, there are 4 errors out of 10 phones w.r.t. the closest reference pronunciation (e.g. QAEDA: /ay/ to /aa/, insert /ey/; SCHIAVO: insert /iy/, /ax/ to /aa/) resulting in a 40% PER.

Since the plex – best was artificially selected for this metric, it becomes the upper-bound (although this isn’t the case for speech tasks shown below). In Figure 4.8 the PER is plotted for each of the methods at 3 system configurations. The 3 configurations were created with different levels of language model pruning, and demonstrate differences based on system performance (in WER). The systems’ WER on the RT04 data set at the various configurations were 29.3%, 24.5% and 19.4% corresponding to 360, 390, and 450 respectively. Note the x-axis is #words, which corresponds to the number of the OOV types that were decoded via the unsupervised method, and hints at a limitation that will be discussed below. With regard to PER, the unsup – reco has lower error rate at all system configurations compared to sup – force, which accords with the results below.

The methods for selecting between candidate pronunciations described above were used to decode 100 hours of speech that contained all of the OOVs. Standard WER was used to compare these methods in Table 2. Note that unsup – reco outperforms all others. Also, note that the candidate pronunciations give

<table>
<thead>
<tr>
<th>word</th>
<th>hyp prons</th>
<th>ref prons</th>
<th>phn err%</th>
</tr>
</thead>
<tbody>
<tr>
<td>QAEDA</td>
<td>k aa d ax</td>
<td>k ay d ax</td>
<td></td>
</tr>
<tr>
<td>QAEDA</td>
<td>k aa ey d ax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCHIAVO</td>
<td>sh ax v ow</td>
<td>s k h aa v ow</td>
<td>40</td>
</tr>
<tr>
<td>SCHIAVO</td>
<td>s k y ax v ow</td>
<td>sh iy aa v ow</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7: Example pronunciations and PER

high confidence to find examples of words like QUEDA. We may not know how many times it was spoken, or where in the audio, but we can still use the entire broadcast to help us choose between hypothesized pronunciations for QUEDA.

In the absence of labeled examples we use unsupervised recognition to select between candidate pronunciations. We decode data likely to contain the OOV with each candidate, calculate the average acoustic likelihood over the entire data, and choose the candidate with the highest average likelihood as the 'winner'. This corresponds to using Equation 1 to decode speech 'as is' (without the extra constraint on the decoding path to the reference as in the supervised case).
Table 4.8: LVCSR WER

<table>
<thead>
<tr>
<th>Method</th>
<th>ASR WER%</th>
<th>#errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>plex-worst</td>
<td>17.8</td>
<td>193,145</td>
</tr>
<tr>
<td>sup-force</td>
<td>17.3</td>
<td>187,772</td>
</tr>
<tr>
<td>unsup-reco</td>
<td><strong>17.3</strong></td>
<td><strong>187,424</strong></td>
</tr>
<tr>
<td>plex-best</td>
<td>17.4</td>
<td>188,517</td>
</tr>
</tbody>
</table>

about a half percent WER range (between the best and worst), and that selecting based on the phone edit distance to the reference does not directly translate to better ASR WER.

4.2.3.2 Spoken term detection

Lattices generated by the LVCSR system for the 100 hours test set were indexed and used for spoken term detection experiments in the OpenFST based architecture described in Parlak and Saraclar (2008b). Our goal was to see whether our WER results correlated with another speech task like spoken term detection. To this end, the same sets of pronunciations were used as queries to the STD system. Results from the OpenFST based indexing system are presented in a DET curve using NIST formulas and scoring functions/tools from the NIST 2006 evaluation. The DET curves in Figure 4.9 show that *plex – best* and *unsup – reco* work the best for detection at nearly all operating points.
4.2.4 Part II - Summary

We have presented an unsupervised method for pronunciation validation via recognition that works better than conventional validation via forced-alignment. This success has been demonstrated using 3 metrics for large-scale speech tasks: Phone Error Rate on a large set w.r.t. a reference lexicon, LVCSR Word Error Rate on decoding a 100 hours of speech, and STD DET Curves on the same.

The usual argument for unsupervised speech methods: they save considerable time and money over speech transcription or using a linguist, which is enticing as long as the performance degradation isn’t too harmful. However, for selecting a candidate pronunciation our unsupervised method does not suffer any degradation, and actually performs better as it naturally filters out unhelpful speech samples by employing the power of comparison (search) and a language model. In all of the experiments our notion of phone errors were based on a word-to-phone pronunciation lexicon; there were no manual phonetic transcriptions used.

There are several limitations to this method. Unsupervised recognition can’t always verify a word (if neither pronunciation is ever decoded), although this provides a natural check against comparing many bad candidates (alignment will always give a score). It requires having seen it or words like it in text (LM), which is not unreasonable given that a word comes into fashion somehow. It’s possible that false alarms might hurt (if an OOV sounds like common word), but the 3 configuration experiments indicate that isn’t a problem for these words of interest. Finally, the performance could depend on amount or type of data decoded, which is the basis of our future work.

4.3 Conclusion

Several tools were used during the course of this workshop. Speech recognition systems were built using the IBM speech recognition tool kit and IBM’s LVCSR decoder was used for generating lattices, N-best and consensus networks to index the audio. An OpenFST based Indexing and Search Tool to be released shortly as open source was developed at Bogazici University and extended during the workshop. This is an extended implementation of (Allauzen, Mohri and Saraclar, 2004). It can handle arbitrary fsm queries, implements term thresholding functions and two-pass and single-pass search for precise times of occurrence of the query terms, and provides rapid indexing and search capabilities. The key directions of research that the results in this workshop point to include, continued work on the multilingual aspect of using web pronunciations, better ways to select pronunciations that reduce false alarms and methods to combine features from ASR (e.g. likelihood scores) and Spoken Term Detection (e.g. scores) in a classifier framework to identify the correct pronunciation of an OOV term.
Chapter 5

Acknowledgments

We wish to thank the sponsors of the 2008 workshop, including the National Science Foundation, Google, IBM, the Human Language Technology Center of Excellence (HLTCOE) at the Johns Hopkins University, and the JHU Center for Speech and Language Processing, for their support of this Workshop.

For the transcription/transliteration subteam, prior work on this project performed at the University of Illinois at Urbana-Champaign was supported by the Department of the Interior under award NBCHC040176 and by a Google Research Award.

Several individuals contributed valuable prior work to the topics investigated in this workshop. These include Su-youn Yoon, Kyoung-kyoung Kim and Tao Tao. Alex Klementiev also provided us with his software for Russian-English transcription.
Bibliography


et al., L. B., 2008. Combination of strongly and weakly constrained recognizers for reliable detection of oovs. In: *Proc. ICASSP.*


Goldwasser, D., Roth, D., 2008. Active sample selection for named entity transliteration. In: *ACL.*


Hieronymous, J., 1993. Ascii phonetic symbols for the world’s languages: Worldbet. URL citeseer.ist.psu.edu/hieronymous93ascii.html


Klementiev, A., Roth, D., 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In: *ACL.*


52


Shao, L., Ng, H. T., 2004. Mining new word translations from comparable corpora. In: COLING.


