Experiments towards a better LVCSR System for Tamil

Melvin Jose Johnson Premkumar[1], *Ngoc Thang Vu*[2], *Tanja Schultz*[2]

[1]Department of Computer Technology, Anna University, Chennai, India
[2]Cognitive Systems Lab, Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT).

melvin.jose_j@yahoo.com, {thang.vu,tanja.schultz}@kit.edu

Abstract

This paper summarizes our latest efforts in the development of a Large Vocabulary Continuous Speech Recognition (LVCSR) system for Tamil at different levels: pronunciation dictionary, language modeling (LM) and front-end. Usually in Tamil there are not many word-pronunciation pairs to train data-driven grapheme-to-phoneme (G2P) converters. Therefore, we explore the correlation between the amount of training data and the performance of the grapheme-to-phoneme (G2P) conversion. To address the morphological complexity of Tamil, we investigate different levels of morphemes for language modeling including a comparison between our Dictionary Unit Merging Algorithm (DUMA) and *Morfessor*, followed by various experiments on hybrid systems using word and morpheme LMs. Finally, we integrate our multilingual bottle-neck features framework with Tamil LVCSR. The final best system produced 21.34% Syllable Error Rate (SyllER) on our Tamil test set.

Index Terms: Tamil LVCSR, morpheme segmentation, language model, hybrid system, multilingual bottle-neck features

1. Introduction

Recently, there has been a dramatic improvement in the performance of speech and language technology with an increasing number of systems being deployed in a large number of applications. Though Tamil is spoken by close to 70 million people in India, Sri Lanka, Singapore, and Malaysia, it has failed to receive the attention that some of the other languages in economically developed countries have been receiving towards the development of speech technology. A handful of the previous works in Tamil include [2], [3], [4], and [5]. Although there has been significant effort to address specific parts of a speech recognition system for Tamil, most of them are only on a small vocabulary set. For example in [6], [7] and [8], the authors investigate various LM configurations for Tamil.

In our previous work [1], we built our baseline Tamil LVCSR system. We aimed to address the morphological complexity of Tamil by proposing a Dictionary Unit Merging Algorithm (DUMA), a word segmentation algorithm which generated merged syllable units (DUMA units). We showed the performance of Tamil LVCSR at three levels - word, syllable and DUMA unit. The SyllER of the three systems were reported as 29.30%, 34.16% and 24.87% respectively. Due to a mistake, we reported 24.87% SyllER with DUMA in [1]. After correcting it, we obtained a SyllER of 28.08% which is still better than the syllable-based system by 17.79% relative. After removing a noisy utterance from the database, the SyllER of the word-based baseline was found to be 27.73%. However, Wilcoxon signed-rank tests at the level of 0.05 (wilc-0.05) show that the superiority of the word-based system over DUMA is not statistically significant. Therefore our motivation was to enhance DUMA. Due to the close performance of word- and DUMA-based systems, we combined both these approaches in hybrid systems i.e. combining words and morphemes in LMs.

In this paper, we present our investigations on Tamil LVCSR at different levels: pronunciation dictionary, language modeling and front-end. First, we explore the correlation between the amount of training data and the performance of G2P conversion for Tamil. Next, experiments on various morpheme level LMs for Tamil are conducted which include a comparison between our approach - DUMA and *Morfessor* [9] and experiments on a hybrid system using word and morpheme LMs. Finally, we apply the multilingual bottle-neck features [10, 11] to Tamil LVCSR.

The remainder of the paper is organized as follows. In Section 2 we describe our speech and language resources. Our G2P experiments are described in Section 3. Section 4 presents our investigations on morpheme level and hybrid ASR systems for Tamil. In section 5, we discuss our experiments using Multilingual bottleneck features. Section 6 concludes our work with an outlook into future work.

2. Tamil Language Resources

2.1. Data Corpus

The Tamil text corpus was crawled using RLAT [12] from popular Tamil news websites and normalized. The websites were crawled with a link depth of 10, i.e. the crawler went recursively 10 levels deep from each level after the level was completed crawled. Table 1 gives the list of websites crawled. The collected text was cleaned and normalized using the following four steps (1) Remove all HTML-Tags and codes, (2) Remove special characters, non-Tamil characters and empty lines, (3) Convert numbers, dates, time and common abbreviations to their equivalent text form, and (4) Remove leading and trailing white spaces and write each sentence on a separate line.

Table 1: *List of crawled websites*

	Website URL	Link depth
1	www.dinamalar.com	10
2	www.dinakaran.com	10
3	www.dinamani.com	10

2.2. Speech Corpus

The speech data for our Tamil recognizer was collected in Tamil Nadu, India in two stages: In the first stage, 68 speakers were each asked to read several lines (ranging between 30 and 300) from Thirukkural, a Tamil literary classic which contains 1,330 verses and is considered to be one of the most important works in Tamil. This accrued to almost 17 hours of speech data. From the data collected in the first stage, data from 5 speakers which added up to 1 hour was separated and designated as the development set. In the second stage, additional speakers were asked to read newspaper prompts collected from the newspapers mentioned in Table 1. A new set of 29 speakers participated in this exercise. The data from this stage accrues to 1 hour and constitutes the test set. All speech data was recorded with a close-speaking microphone and in quiet environmental conditions. A sampling rate of 16 kHz with a resolution of 16 bits was used for the data which was stored in PCM encoding. Table 2 describes our speech corpus.

Table 2: *Speech corpus description*

Set	#Speakers		#Utterances	Duration
	Male	Female		
Training	30	33	1012	15h 50min
Development	2	3	51	1hr 4min
Test	14	15	369	1hr 0min
Total	46	51	1433	17hr 54min

3. Grapheme-to-Phoneme mapping

Tamil language consists of 12 vowels and 18 consonants. Each of the 18 consonants individually combine with the vowels to form 216 additional graphemes. There exists a special grapheme named "aytam" which is neither a consonant nor a vowel. Unfortunately, the G2P conversion task is not very straightforward in the case of Tamil, for the following two reasons: (1) Confusion between allophones p (b), t (d), th (dh), k (g) and c (j) (s) which are very difficult to solve with linguistic rules and (2) the transcription of borrowed words which do not have a standard pronunciation. While most Indian languages are phonetic in nature i.e. they possess a one-to-one correspondence between orthography and pronunciation, Tamil script, although phonetic in nature has a lot of exceptions. Previous work on G2P conversion for Tamil was done in [13] and [14] where the authors explored rule-based and Decision Tree Learning-based approaches.

In [15], the authors conclude that Sequitur [16] and Phonetisaurus [17, 18] perform better than the other existing G2P techniques for LVCSR tasks. For Tamil, both tools give very similar Phoneme Error Rates (PER), however, the training time taken by Phonetisaurus (minutes) was much lower than that taken by Sequitur (hours). Thus, we report the results obtained from Phonetisaurus and explain the algorithm used.

In Phonetisaurus, weighted finite-state transducers are used for decoding as a representation of a graphone-based n-gram LM trained on data aligned by an advanced $M : M$ alignment algorithm [17]. The n-gram can be trained using any standard LM Toolkit in which Kneser-Ney discounting with interpolation is used for smoothing. Decoding is done using OpenFST [19].

From a manually transcribed lexicon of 35k words, we select incremental amounts of data to investigate the correlation between the training size and the phoneme accuracy. The n-gram size of all our models is $N = 7$. For testing the models, we used a test lexicon with 5k words which was handcrafted by native speakers. Our best G2P model achieves a phoneme accuracy of 99.05% on the test lexicon. This model is used to generate the pronunciation lexicons for our experiments. Figure 1 correlates the training data size and the phoneme accuracy.

It should be noted that the subsets of the 35k train lexicon are selected automatically by a program if all the phonemes occur a minimum number of times in the subset. The 10k subset performs worse than the 3k subset since the program selects a poor set of word-pronunciation pairs. From Figure 1 it can be seen that for the G2P performance to cross the 98% mark, close to 5000 carefully selected word-pronunciation pairs are required.

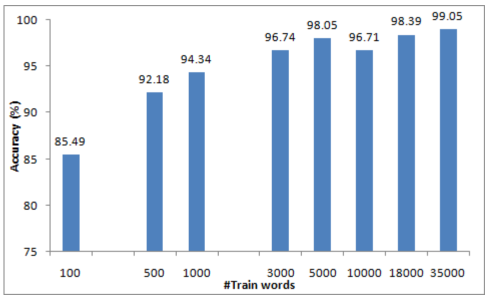


Figure 1: *Training size vs Accuracy*

4. Morpheme-based LMs for Tamil

4.1. Introduction

Tamil is a member of the Dravidian Language family which is one of the morphologically rich families of languages comparable to Finno-Urgic languages and Turkish. This morphological complexity often causes data sparsity issues and results in high OOV-rates and LM perplexities. A traditional approach to overcome this problem is to use a very large vocabulary. Using a very large search vocabulary also leads to high OOV rates and high resource requirements such as CPU time and memory. Alternatively, morpheme-based LMs can be used to lower the OOV rate and decrease the perplexity, reduce the resource requirements and achieve better accuracy. Normally, morpheme generation is carried out by applying morphological decomposition to words based on supervised or unsupervised approaches. Supervised approaches like in [20] use linguistic knowledge in the form of a set of manual rules to perform Tamil word decomposition. Other supervised methods make use of lexical and syntactic knowledge like in [21, 22, 23]. The disadvantage of the supervised techniques is the need of expert knowledge and hours of work. On the other hand, the unsupervised approaches are statistical data driven approaches like the algorithm in [1] which is based on the frequency of pronunciation transitions and [9], where the authors present an unsupervised methods based on Minimum Description Length (MDL). The unsupervised approaches are language independent and require no prior linguistic knowledge.

There have been a few previous works on the effect of morpheme-based LMs for Tamil. [7] explores the effect of various LMs for Tamil, concluding that morpheme-based LMs are better than the others. However, an supervised morphological analyzer [14] was used in the experiments on a small vocabulary. In [8], a morpheme-based LM based on [25] has been built and is found to be better than the word-based LM. However, hybrid systems - combining words and morphemes have not yet been tried for Tamil. In this section we examine various unsupervised morpheme level and hybrid LMs for Tamil.

4.2. DUMA-derived morpheme LM

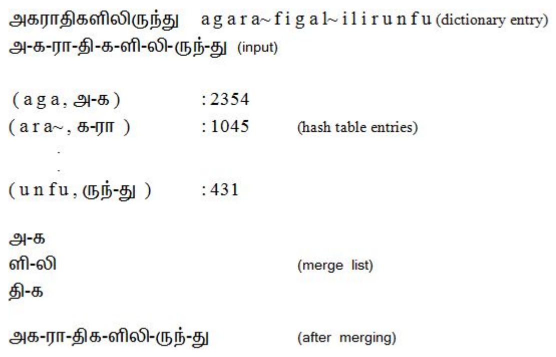
In our previous work [1], better than the algorithm proposed in [26], a data-driven, statistical approach that requires no a-priori linguistic knowledge called DUMA was proposed. It aims to determine appropriate dictionary units for Tamil, to overcome the high OOV rate and LM perplexity due to the rich morphology of Tamil.

The inputs to the algorithm are a pronunciation dictionary, the LM training text and a vowel list. The vowel list is the only linguistic knowledge required by the algorithm. Initially, we segment the entire text into syllables using the syllabification algorithm stated in [4]. We also include word boundary information in the syllabified text i.e. we prepend a '-' to every syllable that does not occur at the start of a word. Then we obtain all possible syllable pairs from the syllabified text. Each possible pair is then looked up in the dictionary and the pronunciation of the vowel-vowel transition is retrieved.

The merging algorithm is governed by the following iterative steps:

1. First, we compute a hash table that maps the vowel-vowel transition, the corresponding syllable pair to the frequency of the pair in the LM text.
2. For each vowel-vowel transition in the hash table, we place the most frequent syllable pair into a merge-list.
3. We merge all the pairs in merge-list in the segmented corpus.

We only merge pairs that occur within a word, and chose not to merge pairs across word boundaries since Tamil has a fixed word boundary. We use the merge-list obtained after step 2 of the unit merging algorithm to merge both the training and test transcripts. Figure 2 shows the various stages of DUMA.



அகராதிகளிலிருந்து a g a r a~ f i g a l~ i l i r u n f u (dictionary entry)
அ-க-ரா-தி-க-ளி-லி-ருந்-து (input)

(a g a , அ-க) : 2354
(a r a~ , க-ரா) : 1045 (hash table entries)
.
.
(u n f u , ருந்-து) : 431

அ-க
ளி-லி (merge list)
தி-க

அக-ரா-திக-ளிலி-ருந்-து (after merging)

Figure 2: *Various stages of DUMA*

4.3. *Morfessor*-derived morpheme LM

A very popular unsupervised morphology learning technique is the *Morfessor* [9]. It uses the MAP (Maximum a *Posteriori*) algorithm to find the most likely morpheme boundaries. Each word in the lexicon of the corpus is recursively broken down into all possible segmentations. Given the lexicon of morphs (M), lexicon of words (W), each of which has n segments, μ is a morph with a frequency function f_μ and a length function s_μ. The model is defined as below:

$$P\left(corpus\,|M\right) = \prod_{j=1}^{W}\prod_{k=1}^{n_j} P\left(\mu_{jk}\right) \tag{1}$$

$$P\left(lexicon\right) = M! * P\left(f_{u1}\dots f_{uM}\right) * P\left(s_{u1}\dots s_{uM}\right) \tag{2}$$

$$P\left(M\,|corpus\right) = P\left(corpus\,|M\right) * P\left(lexicon\right) \tag{3}$$

In our experiments, we learn morphology based on word types rather than tokens since it has been seen in [9] that linguistically best segmentation is obtained by learning from word types. The resulting segmentation obtained is recursively used to segment the lexicon thereby performing an Expectation-Maximization (EM) using the Viterbi algorithm on the morph segmentations. It was noted that perplexity of the LM built using the generated morphemes converged after 11 iterations of EM in our case. *Morfessor* generates unrealistic segments on unicode text since it segments randomly at any length irrespective of the letter present. This sometimes segments compound Tamil unicode characters into characters which are not defined in Tamil. A program is made to sweep through all the segmentations to rectify this problem and delete spurious segments. We use SRILM Toolkit [27] for building all our LMs.

Table 3: *DUMA vs. Morfessor*

System	LM Order	PPL	OOV rate (%)	SyllER (%)
Word baseline	3	5,780	4.9	27.73
DUMA	3	14,344	1.5	28.08
	4	11,521	1.5	27.65
Morfessor	3	10,377	0.8	28.49
	4	9,175	0.8	28.66

For a fair comparison across all models, the perplexity and OOV rates are normalized by the number of words present as in [21] and [22]:

$$PPL^* = (PP)^{N_d/N_{org}} \tag{4}$$

$$OOV_{norm} = OOV * N_d/N_{org} \tag{5}$$

where PPL^* and OOV_{norm} denote the normalized perplexity and OOV rate, N_d is the number of morpheme tokens in the decomposed data, and N_org is the number of original words.

The lower perplexity of the word-based system is due to its high OOV rate which ignores many rare words from the perplexity calculations. The shrinkage of the LM span in the morpheme based LMs might contribute to their high normalized perplexities. From Table 3, we see that the DUMA system performs better than the *Morfessor* system (statistically significant by wilc-0.05). *Morfessor* produces morpheme units with an average length of 2.41 syllables and on average 2.16 morphs per word compared to an average length of 2.01 syllables for DUMA units and 2.36 units per word produced by DUMA. The production of longer and lower number of units by *Morfessor* as compared to DUMA might be the reason behind its bad performance.

4.4. Hybrid System: Full words + morphemes

4.4.1. Motivation

The main aim of formulating DUMA was to obtain a trade-off between small syllable units and agglutinative word units. However, it can be seen that DUMA also suffers from acoustic confusabilities due to short units. Hence, we chose not to merge the top Nk words of the vocabulary and segment only the remaining vocabulary. Using full words in Morpheme LMs was found to be useful in previous experiments with Arabic [22] and German [28]. This has the following advantages:

- It reduces the acoustic confusability caused by short DUMA units.
- It increases the range of the acoustic and language model.
- The presence of the top Nk words during the training is found to help the system.

4.4.2. Experiments

We chose the word-based system and the DUMA system from Table 3 as our baseline systems to build the hybrid system. The word- and DUMA-based LMs have 40M and 60M tokens, 602k and 223k types. Starting from the DUMA system, we gradually increased the number of words that would not be segmented. Since morphemes and words are of different lengths, their optimal performance may occur at different n-gram orders [9]. Hence we also experiment with 4-gram LMs in addition to 3-gram LMs for all the morpheme-based systems. Table 4 summarizes all our experimental results.

Table 4: *Comparison of Hybrid Systems (WB: Word-based, DB: DUMA-based, H: Hybrid, mrfs: morphemes, wrds: words)*

Sys.	#mrfs	#full wrds	OOV rate (%)	3-gram		4-gram	
				PPL	SyllER (%)	PPL	SyllER (%)
WB	0k	602k	4.9	5,870	27.73	5,467	27.65
DB	223k	0k	1.5	14,344	28.08	11,521	27.65
H	221k	5k	1.5	14,240	27.74	12,024	27.70
	222k	15k	1.5	16,631	27.73	14,838	27.58
	219k	20k	1.5	12,008	**26.55**	10,861	26.76
	212k	25k	1.5	11,906	26.83	10,661	26.80

The high values of the perplexities in the above table are since they are normalized as in Eq. 4. To the DUMA baseline (3gram) and the best hybrid system (219k #mrfs & 20k #wrds), we add another 4.9M lines of text and build two new LMs. The perplexity of these LMs on the test set are 12,804 and 5,966 and the SyllER are 27.37% and 25.73%. Thus with the additional text, the hybrid system (219k #mrfs & 20k #wrds) significantly outperforms the DUMA baseline and is used in our bottle-neck experiments.

5. Multilingual Bottle-neck features

In the last few years, the use of multi layer perceptron (MLP) for feature extraction showed impressive ASR performance improvements. In many setups and experimental results, MLP features proved to be of high discriminative power and very robust against speaker and environmental variations. Hence, in this paper we integrate these features into our Tamil ASR system. Figure 3 shows the layout of our MLP architecture. As input to the MLP network, we stacked 11 adjacent MFCC feature vectors and used phones as target classes. A 5 layer MLP was trained with a 143-1500-42-1500-81 feed-forward architecture. In the pre-processing of the Bottle-Neck (BN) systems, the LDA transform is replaced by the first 3 layers of the MLP using a 143-1500-42 feed-forward architecture (BN), followed by stacking of 5 consecutive output frames. Finally, a 42-dimensional feature vector is generated by an LDA, followed by a covariance transform. All neural networks were trained using ICSI QuickNet3 software [29].

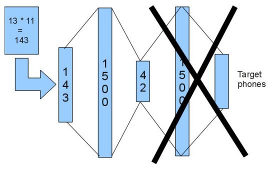


Figure 3: *Bottle-Neck feature*

However, training an accurate MLP for a new language with a small amount of data is not a trivial task. In [10, 11], we showed that multilingual MLP (ML-MLP) is a good initialization for MLP training especially for a new language and therefore, we could train an MLP for a new languages such as Creole and Vietnamese. Figure 4 illustrates the initialization scheme. For the new language, we select the output from the ML-MLP based on the IPA table and use it to intialize the MLP training. All the weights from the ML-MLP are taken but only the output biases from the selected targets are used.

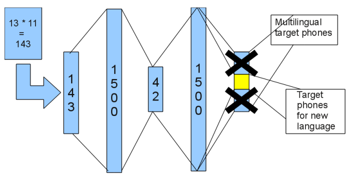


Figure 4: *Initialization for MLP training or adaptation using a multilingual MLP*

In this paper, we apply the multilingual MLP which was trained with English, French, German, and Spanish [10, 11] to initialize the MLP training for Tamil. Table 5 shows the frame-wise classification accuracy of all the MLPs (one with random initialization and the other with multilingual MLP initialization on the cross-validation data) and the SyllER on the Tamil evaluation set. Using ML-MLP for initialization of MLP training, we obtain our best performance with 21.34% SyllER on the eval set.

Table 5: *Frame accuracy on cross validation set of MLP training and SyllER on the Tamil evaluation set*

Systems	AccCV(%)	SyllER(%)
Baseline	-	25.73%
Random-init	69.85	25.03%
Multilingual-init	76.1	21.34%

6. Conclusions and Future Work

In this paper, we present our investigation on Tamil LVCSR at different levels: front-end, dictionary and language model. Firstly, we explored the correlation between the amount of training data and the performance of the G2P conversion for Tamil. Secondly, we investigated various morpheme level systems for Tamil: we showed that our DUMA system slightly outperforms the *Morfessor* system. However, a hybrid system with morphemes extracted from DUMA and unsegmented top 20k words produced the most improvement of about 4% and 5% relative SyllER compared to our word-based and DUMA-based baseline systems. Finally, we integrated multilingual bottle-neck features to Tamil LVCSR and obtained an additional 18% relative improvement in SyllER. The best system obtained 21.34% SyllER on the Tamil evaluation set. From previous work, it can be seen that a comprehensive LVCSR system for Tamil has not yet been developed. We believe that our work is among the first to build large vocabulary systems for Tamil. In the future, we would like to investigate the effect of using high order n-grams in our approaches. We would like to build Factored Language Models (FLMs) [30], which were found to be useful for other morphologically rich languages like Arabic, for Tamil. The use of *graphones* as units in the Hybrid LMs could also be promising.

7. Acknowledgments

The authors would like to thank Dr. Sophia for help in transcribing the training and test lexicon for the G2P experiments.

8. References

[1] J. Melvin Jose, N.T Vu, and T. Schultz, "Initial Experiments with Tamil LVCSR", in Proceedings of IALP, 2012.

[2] R. Kumar, S. Kishore, A. Gopalakrishna, R. Chitturi, S. Joshi, S. Singh, R. Sitaram, "Development of Indian language speech databases for large vocabulary speech recognition systems", in Proceedings of SPECOM, 2005.

[3] M. Plauche, N. Udhyakummar, C. Wooters, J. Pal, and D. Ramachadran, "Speech Recognition for Illiterate Access to Information and Technology", in Proceedings of First International Conference on ICT and Development, 2006.

[4] A. Lakshmi and H.A. Murthy, "A syllable based continuous speech recognizer for Tamil", in Proceedings of Interspeech, 2006.

[5] R. Thangarajan, A.M. Natarajan, M. Selvam, "Word and triphone based approaches in continuous speech recognition for Tamil language", WSEAS Trans. Sig. Proc, 4(3):76-85, 2008.

[6] S. Saraswathi, T.V. Geetha, "Design of language models at various phases of Tamil speech recognition system", International Journal of Engineering, Science and Technology, 2(5):244-257, 2010.

[7] S. Saraswathi and T.V. Geetha, "Building Language Models for Tamil Speech Recognition System", LNCS. Volume 3285, pp 161-168, 2004.

[8] S. Saraswathi and T.V. Geetha, "Morpheme based language model for Tamil speech recognition system", The International Arab Journal of Information Technology. Vol. 4, No. 3, July 2007.

[9] M. Creutz, T. Hirsimaki, M. Kurimo, A. Puurula, J. Pylkkonen, V. Siivola, M. Varjokallio, E. Arisoy, M. Saraclar, and A. Stolcke, "Morph-based speech recognition and modeling of out-of-vocabulary words across languages", ACM Transactions on Speech and Language Processing, vol. 5, no. 1, Dec. 2007.

[10] N. T. Vu, F. Metze and T. Schultz, "Multilingual bottleneck features and its application for under-resourced languages", in Proceedings of SLTU, 2012.

[11] N.T Vu, W. Breiter, F. Metze, and T. Schultz, "An Investigation on Initialization Schemes for Multilayer Perceptron Training Using Multilingual Data and their Effect on ASR Performance", in Proceedings of Interspeech, 2012.

[12] N. T. Vu, T. Schlippe, F. Kraus, and T. Schultz, "Rapid Bootstrapping of five Eastern European Languages using the Rapid Language Adaptation Toolkit" , in Proceedings of Interspeech, 2010.

[13] C. S. Kumar, V. Shunmugom, U. Nallsamy and R. Srinivasan, "Rule-based automatic grapheme to phoneme conversion for Tamil", in Proceedings of ICSLT, 2004.

[14] U. Nallasamy, C. S. Kumar, R. Srinivasan and R. Swaminathan, "Decision tree learning for automatic grapheme to phoneme conversion for Tamil", in Proceedings of SPECOM, 2004.

[15] S. Hahn , P. Vozila , M. Bisani, "Comparison of Grapheme-to-Phoneme Methods on Large Pronunciation Dictionaries and LVCSR Tasks", in Proceedings of Interspeech, 2012.

[16] M. Bisani, H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion", in Speech Communication, 50(5):434-451, 2008.

[17] J. Novak, D. Yang, N. Minematsu, K. Hirose, "Initial and Evaluations of an Open Source WFST-based Phoneticizer", The University of Tokyo, Tokyo Institute of Technology.

[18] D. Yang, et. al., "Rapid development of a G2P system based on WFST framework", ASJ 2009 Autumn session, pp. 111-112, 2009.

[19] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: a general and efficient weighted finite-state transducer library", Prague, Czech Republic, pp. 11-23, 2007.

[20] P. Anandan, K. Saravanan, R. Parthasarathi, and T.V. Geetha, "Morphological Analyzer for Tamil", in Proceedings of ICON, 2002.

[21] K. Hacioglu and B. Pellom and et. al., "On Lexicon Creation for Turkish LVCSR", in Proceedings of Eurospeech, 2003.

[22] A. El-Desoky, C. Gollan, D. Rybach, R. Schluter, and H. Ney, "Investigating the use of morphological decomposition and diacritization for improving Arabic LVCSR", in Proceedings of Interspeech, 2009.

[23] W. Byrne, J. Hajic , P. Ircing, P. Krbec, and J. Psutka, "Morpheme based language models for speech recognition of Czech", in Text, Speech and Dialogue, ser. LNCS, vol. 1902, pp. 139 - 162, 2000.

[24] M. Adda-Decker, "A corpus-based decompounding algorithm for German lexical modeling in LVCSR", in Proceedings of Eurospeech, 2003.

[25] J. Kneissler, and D. Klakow, "Speech recognition for huge vocabularies by using optimized sub-word units", in Proceedings of Eurospeech, 2001.

[26] D. Kiecza, T. Schultz, and A. Waibel, "Data-Driven Determination of Appropriate Dictionary Units for Korean LVCSR", in Proceedings of ICASSP, 1999.

[27] A. Stolcke, "SRILM - An Extensible Language Modeling Toolkit", in Proceedings of ICSLP, 2002.

[28] A. El-Desoky, M.A. Basha Shaik, R. Schluter, and H. Ney, "Morpheme Based Factored Language Models for German LVCSR", in Proceedings of Interspeech, 2011.

[29] http://www.icsi.berkeley.edu/Speech/qn.html

[30] K. Kirchhoff, D. Vergyri, J. Bilmes, K. Duh, and A. Stolcke, "Morphology-based language modeling for Arabic speech recognition", in Proceedings of ICSLP, 2004.