THE JOURNAL OF THE ACOUSTICAL SOCIETY OF KOREA VOL.26, NO.1E 2007. 3 pp. 1-7

A Study on the Language Independent Dictionary Creation Using International Phoneticizing Engine Technology

Chwa-Cheul Shin*, In-Sung Woo,* Heung-Soon Kang*, In-Soo Hwang*, Suk-Dong Kim*
*Department of Computer Science, Hoseo University
(Received October 19 2006; Revised December 27 2006; Accepted February 7 2007)

Abstract

One result of the trend towards globalization is an increased number of projects that focus on natural language processing. Automatic speech recognition (ASR) technologies, for example, hold great promise in facilitating global communications and collaborations. Unfortunately, to date, most research projects focus on single widely spoken languages. Therefore, the cost to adapt a particular ASR tool for use with other languages is often prohibitive. This work takes a more general approach. We propose an International Phoneticizing Engine (IPE) that interprets input files supplied in our Phonetic Language Identity (PLI) format to build a dictionary. IPE is language independent and rule based. It operates by decomposing the dictionary creation process into a set of well-defined steps. These steps reduce rule conflicts, allow for rule creation by people without linguistics training, and optimize run-time efficiency. Dictionaries created by the IPE can be used with the Sphinx speech recognition system. IPE defines an easy-to-use systematic approach that can lead to internationalization of automatic speech recognition systems.

Keywords: *ASR, Unicode, phonetics, orthography, lexicon, phoneme*

I. Introduction

Many interesting questions arise when adapting existing speech recognition systems to languages other than the original target language [1]. The core design of these systems involves many assumptions that do not hold when applied to other languages. For example, languages often use different writing systems, phoneme sets, and rules of pronunciation. Often, adequate performance requires significant tuning over time by experts.

In order for a speech recognition system to address additional languages, a number of approaches have been proposed [2]. Some advocate rebuilding systems from scratch to create new systems that uniquely suit the target languages [3]. Others prefer rebuilding statistically based systems aimed at cross language portability [4]. Some systems that rely on machine learning [5] are dependent on the amount and quality of existing data, an assumption that doesn't hold for many languages.

Although these approaches have significantly contributed to speech recognition technology, they are very costly in that they require linguistics expertise and extensive redundant development. Globalization necessitates rapid deployment; therefore an approach that is perhaps better suited is one that makes the most use of existing systems. The DIPLOMAT project [6] takes this approach and successfully adapted its core system for the

Corresponding author: Suk-Dong Kim (sdkim@hoseo.edu)
School of Computer Engineering, Hoseo University 29-1 Sechul-Ri Baebang-Myun Asan-City Choongnam 336-795, Korea

Serbo-Croatian, Haitian Creole, and Korean languages.

Our approach addresses two important factors required in any realistic attempt at generalized multilingual ASR. The first of these issues is user friendliness. Our simple design provides a user-friendly environment enabling non-linguistically trained native speakers to utilize the ASR tools. We rely on the availability of a native informant and utilize their effective knowledge of the language, but that person doesn't require formal training in linguistics or speech recognition. This approach is different from prior attempts to create language independent systems [7, 8]. Those attempts generally do not utilize character sets that are familiar to most native speakers, neither do they offer grammars that are legible by non-linguistically trained experts.

The second factor that we address is the need for a step-by-step language independent phoneticizing process. Because we want to process an international language as specific engine (IPE) and want to make the standard PLI, we deal with separate processing. In speech technology terms, a language is unique in the way it sounds and in its script, both of which can be found in its lexicon. The lexicon is the most localized part of any speech system, since once we convert speech data to a common character set, many of the other components of any system need no further internationalization.

This paper describes our language independent phoneticizing process that generates a lexicon useable for speech recognition. This process consists of the following four steps. These are (a) Transliterating codepoints from Unicode, (b) Phonetically standardizing rules, (c) Implementing grapheme to phoneme rules, and (d) Implementing phonological processes. The application of these steps takes a Unicode string as input and produces a corresponding phonetic string, solving the common character set issue along the way. The discrete decomposition of the phoneticizing process reduces rule collisions and achieves sequential rather than global rule applications, an approach which is significantly more computationally feasible.

This paper is organized as follows. Section 2 describes our process to create lexicons useable by speech recognition systems. Section 3 describes the syntax of the Phonetics Language Identity (PLI) grammar that we use. Section 4 briefly describes our initial implementation of the International Phoneticizing Engine which interprets the PLI grammar. Section 5 describes English and Korean lexicons that we create to integrate with Carnegie Mellon's Sphinx Speech Recognition System [9]. Section 6 evaluates the results of these efforts. Section 7 describes our future goals and concludes the paper.

II. The Four Step Phoneticizing Process

Rule collisions are a major obstacle to successful rule-based phoneticizing of a language [4]. For this reason, we divide the process into four well-defined steps. These step process consists of a Unicode transliteration followed by normalization of the orthography, phoneticization of the normalized string, and finally phoneme clarification. The resulting set of phonemes is then integrated into an ASR system. Figure 1 illustrates this basic scheme.

The work of phoneticizing a language consists of creating four rule sets. Unlike machine learning-based approaches [10], our ultimate aim is not to completely automate the lexical acquisition process, but rather to structure it in a way that will allow native speaker (not necessarily a linguist but computer literate) to make speech technology multilingual. We will take a closer look at each of the four steps in the following subsections.

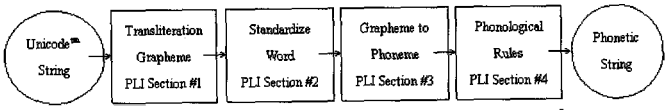

Figure 1. Language Independent Phoneticization in Four Step

Step 1: Transliterating from Unicode to ASCII. Unicode is now the accepted standard universal character set. Unicode is a fixed size character set, where each text element is encoded in 16 bits (UCS-2), which achieves uniformity across languages. More importantly, the Unicode consortium [11] has set standards for the processing of many scripts beyond the capabilities of ASCII (i.e. Hangul syllable decomposition/composition). For these reasons, Unicode must be included in any attempt at language independent lexical acquisition.

The goal of the transliteration step is to map each target language Unicode code point to an ASCII string. Transliteration allowsus to create our own, string-based

internal character, well suited for phonetic processing and conforming to the requirements of existing ASCII based ASR systems.

Transliterating allows us to extract additional information contained in text elements. We use to recognize korean speech using the Sphinx 3 engine, so we work three steps to process korean character. The first step is converting from ascii code to unicode. Next, converting from unicode to intermediate code. The last step convert from intermediate code to internal code. ASCII's original design suits American English and contains a single code point for each letter of the Latin alphabet. This is inadequate for some languages where one text element maps to more than one linguistic phenomenon. For example, in French vowels often carry diacritical marks. In Hangul each text element is a syllable, and a text element may carry up to four jamos (that is, phones). The complexity of the transliterating step varies across languages. It is a trivial step for phonetic languages with few characters; it is more involved for languages with extensive ideographic scripts.

Step 2: Standardizing the orthography. Languages carry in their orthography complexities because they evolve over time. Often the orthography to sound relationship is counter-intuitive (i.e. English: "knight" sounds more like "nite".) Some languages allow multiple spelling orthographies for the same word (i.e. Haitian Creole "pwezidan" and "presidan"). Homophomes (i.e. English "know," "no") force the orthography to mark semantic differences.

These examples illustrate the need for phonetic standardization so speech technology systems can establish important script to sound relationships. As in step 1, the goal of this step is intuitive and self-explanatory, so a non-linguistically trained native speaker can perform it.

Step 3: From Graphemes to phonemes. With a standardized orthography, this step implements basic grapheme to phoneme mapping, and addresses remaining context dependent pronunciation combinations. Phoneme interaction such as nasalization need not be created here in order; those complexities fall to step 4.

Step 4: Phonological processes. Pronunciation rules often depend on the complex interactions caused by creating sounds using the vocal tract. Step 4 differentiates between allophones, depending on their phonetic context. Step 4 also eliminates remaining redundancies. For example, in French, when some identical phonemes are repeated, only one is pronounced (i.e. "tourette": T UW R EH T T T UW R EH T).

III. Phonetic Language Identity (PLI) Grammar

This section describes the grammar that we use to drive the four step process described in the previous section. This grammar is in keeping with our goal of simplicity, modeling the phoneticizing process using locally simple transformations. Users create text files using the PLI grammar syntax.

PLI text files contain four sections. Each section is separated by keywords "#1", "#2", "#3" and "#4" on a line by themselves. All sections need to be present in the file and in order. Each section contains a set of rules which correspond to one of the discrete steps in transitioning from UnicodeTM to phonetics. The PLI syntax is the same for all sections: source string [tab character] target string. PLI processing maps the input source strings to output target strings. The target strings of one section are inputs to the next section. All characters following the characters, //, are treated as comments. We recommend using comments to improve readability. The following subsections describe the rules created for each of the four sections in more detail.

Section #1: Each PLI line contains a UnicodeTM code point in hexadecimal uppercase followed by a tab character and the ASCII transliteration code. We refer to the Unicode code point explicitly (instead of its 16 bit representation) because this allows the PLI format to be in standard ASCII, yet makingit able to refer to the Unicode code point. We recommend users add a comment (starting with //) containing the actual Unicode text element after each PLI statement; this allows a degree of verification. Note that this does not compromise the assumption that the PLI format is ASCII compliant during processing. Example PLI Statement: D4AD {pVt} //팔

Section #2: These lines contain a transliteration character string, a tab character, and a standardized transliteration string. The transliteration character strings

Table 1. Grouping Variable Syntax int PLI

```
(consonants) = {b c d f g h j k l m n p q r s t v w x y z}
//Grouping consonants in one variable

(consonant-sound) = {B K D F G HH JH K L M N P K R S
T V W X Y Z }
//groups consonants sounds in one variable

(consonant) [IY] (consonant-sound)
//use the grouping variables in a rule
```

are produced by section #1 processing. To map the character 'kn' in knight to 'n', we use the PLI statement: kn n

Section #3: Rules in this section contain the standardized transliteration strings generated in section #2, a tab character, and a phoneme string. Two such rules are: th [TH] and i [AY]

Section #4: These rules contain the phoneme sequence strings generated in section #3, a tab character, and a final phoneme sequence string. They often eliminate redundancies. Examples: [G][G] [G] and [X] [K][S]

Reserved Characters are used when creating a PLI files. For example, we recommend the use of encasing marks ([]{}) to enclose single phonetic units (i.e. a phoneme AY expressed as [AY], a Hangul syllable 한 expressed as {han}), although discretion in this matter is left to the user. The PLI grammar also makes use of two reserved characters, "+" and "#." The "+"character (space marker) represents inter-word spaces. The "+" allows the PLI to function both as an exception dictionary and a rule based grammar system. Example: s+i [Z]+ [ay]. Rules that contain the "#" character (the Null phoneme) express unpronounced sequences. Example: [HH]+ #+

Grouping Variables define enumeration vectors of syntactical units that can later be referenced as a class. Using grouping variables eliminates having to explicitly list each unit individually in PLI rules. Users declare a grouping variables by enclosing a variable name of choice within the '<'and'>' symbols. The partial PLI file shown in Table 1 declares two grouping variables, <consonants> and <consonant-sound>. The declaration of a grouping variable must always precede its use (see <consonants> and <consonant-sound> in Table 1.)

Individual elements in two enumeration vectors normally must have a one-to-one mapping between them because the physical positions in the enumeration drive the PLI translation mapping. The Table 1, the <consonants> and <consonant-sound> variables have the same number of enumerated elements. The consonant c is mapped to K. One exception to this one-to-one rule is when the target does not contain any variables. In this case the entire expanded source will map to the same target unchanged. Examples : o<variable> o# and o<variable> o

Since grouping variables are user-defined, PLI syntax allows all levels of sophistication. A non-linguist might just group consonants and vowels together, while a linguist could create grouping variables for many phonetic features. The flexibility is powerful enough to accommodate both linguist and non-linguist users.

IV. International Phoneticizing Engine(IPE)

The IPE, written in Java, is a cross platform interpreter of the aforementioned PLI format. It is a command line application that requires a Unicode text file and the PLI file relevant to the language used in that Unicode file. The output is a phonetic lexicon that is useable by speech recognition systems.

The IPE adheres closely to the syntax and detects any inconsistencies of the PLI it interprets as it encounters them. Users can specify any or all of the four processing steps, and can process the PLI in trace mode. A simple to read output conveniently allows verification of the results.

The IPE sorts all rules in each section in descending order by the length of the source string. Rules with equal source string length retain their original order. Sorting rules permits the more drastic rules to process first, with other rules to follow. Rules are applied in a left to right manner.

V. Creating with IPE Generated Lexicons

To test our IPE approach, we created a variety of phonetic lexicons and integrated them with the CMU Sphinx Speech Recognizer. In order to create PLI files,

the most frequent words of our text corpus were chosenbecause most pronunciation phenomena are encountered in that subset. Secondly, we believe that using high frequency words produces better results when applied to natural text. We describe the PLI creation process in this section.

Korean (Hangul). Two native speakers of Hangul created this PLI file. Korean presents an interesting challenge because it has a very large and very different character set from ASCII, yet it uses a phonetic script requiring further rules. 900 words were used in building the PLI.

In section #1, the IPE mapped each Unicode Hangul phonetic Syllable to a corresponding string of three jamos. The PLI section #1 file contained 11,179 code points that the IPE automatically mapped in ten minutes. Section #2 contains 240 complex rules. Because Hangul is a regular language, each jamo's pronunciation depends on its position in the 3-jamo wide syllable. There are some inter syllable interactions that affect the pronunciation and the corresponding PLI rules were expressed here. In particular, some composed jamo (where one jamo represents the ligature of the two other) have pronunciations entirely dependent on their context. Section # 3 only contains one rule that implements three possible pronunciations depending on the position of the jamo. Section #4 implemented ten nasalization and deletion rules.

English. A native speaker and a fluent speaker created the PLI file. English (expressed in ASCII) offers the easiest language to import from Unicode. Section #1 required only fifty four rules. TheSection #2 PLI file was a challenge because English contains so many pronunciation exceptions. 220 exceptions and 286 standardizing rules required 25 man-hours of work. This effort illustrates the need for a better rule collision detection mechanism. Section #3 contains only forty seven rules due to the large amount of standardization accomplished in section #2. In section #4, we inserted rules that took care of "-ed" endings, and many rules that removed unpronounced letters.

In general, we recommend usingPLI section #2 for rewriting a word in an orthography that is more phonetically correct. For most languages, this approach works well. While developing a PLI for English, we altered this use to also generate phonemes in that same section (of the 286 standardizing rules, 47.5% contains phonemes). The phonemes we produce in this section expresses some sounds using graphemes. Using section #2 in this way sometimes works well, but can often produce side effects that are not desirable. Consider the following four rules:

```
Rule #1: tion     shan
Rule #2: ns+      nz+
Rule #3: ty+      tee+
Rule #4: rite r   ayt
```

The IPE maps the word "portions" to "porshans" using rule #1, and then to "porshanz" by rule #2. But "porshanz" is indeed a decent graphemic for the phonetic end result, "[P][AO][R][SH][AH][N][Z]".

Now consider the following words "majority" and "write." After applying rule #3, the two words respectively become "majoritee" and "write." But after applying Rule #4 our two words respectively become "majorayte" and "wrayt." This is incorrect. This problem can be overcome by modifying rule #3 and rule #4 and introducing some phonetic symbols (ty+ t[IY]+ // Modified rule #3, ite ayt // Modified rule #4). Using these rules, the words become majorit [IY] and wrayt, and leads to a correct phonetic representation.

Other valid approaches are also possible. For example we could add the words to an exception list. Unfortunately, this practice will result in PLIs with long list of word exceptions affecting only the word they express without contributing to the phoneticization process of unexpected words in the target language.

VI. Experimental Results

In this section we analyze the results obtained when we integrated our phonetic lexicons with the Sphinx Speech Recognizer. Table 2 shows results from comparing our English recognizer with a hand tuned dictionary which required several years of work by experts to create. This experiment utilized a dictionary of 2997 words, 68

Table 2. Error Rates using English PLI with SPHINX ASR System, Vs. CMU dictionary 0.5b

Phonetic Acoustic Models	Word Error Rates (%)
CMU Dictionary 0.5b	12.11
IPE English:	30.72
IPE English with minimal changes *	20.07

*10 entries were changed (3 exceptions {kansas, saint, arriving} were added and 7 high frequency words {the, to, what ,a for, from, are} were allowed alternate pronunciations.)

speakers, and 136 utterances. Our scheme achieved an initial error rate of 30.72%. Considering the non-cost of the approach (in time and expenditure), this was a very good starting point. With minimal expert fine-tuning, we modified 10 PLI rules to reduce the error rate significantly to 20.07%. Although this still is far worse than the hand tuned results, it demonstrates that results can be dramatically improved with little effort once the PLI is created.

We next applied our approach to the Korean language using Sphinx combined with the CMU-Cambridge Language modeling Toolkit [12]. The text corpus used for both the language model and the speech data collection corpus was obtained from publicly available online sites. The text, obtained in the KSC Wansung encoding, was converted to Unicode (UCS-2) and broken into sentential utterances. After the utterances were phoneticized, we used a minimum preserving scheme to extract a diphonically rich subset for the recording script. The training data consisted of 21 hours of speech read by 162 (70 female and 92 male) native Korean speakers. The pronunciation dictionary was generated with the aforementioned Korean PLI interpreted by the IPE; it was used for both the training of the acoustic models and the recognition tests. The speakers used in the recognition run (1 female and 1 male) were not included in the training. The test corpus contains 100 utterances and (13.11 minutes of speech).

Table 3 shows the results of our Korean experiment. It would be desirable to compare the robustness of our newly created Korean acoustic model, with our established English acoustic model. However, cross-language ASR system comparisons are difficult. Nevertheless, results show that the Korean acoustic models perform similarly to the English model. Since the Korean acoustic model was built using a pronunciation dictionary automatically created

Table 3. Error Rates using Korean PLI

LM text size: 14358 Dictionary size: 8550 words	Trigram Perplexity: 6.25 Entropy: 2.64 bits	Bigram Perplexity: 38.48 Entropy: 8.40 bits	Unigram Perplexity: 1895.80 Entropy: 10.89 bits
Word Error Rate (%)	8.45	15.67	25.25
Syllable Error Rate (%)	5.54	9.70	16.61

using PLI files and the English acoustic models were conventionally trained with a handcrafted dictionary, we believe that this supports our claim that our process is a viable alternative to handcrafted lexicons.

VII. Conclusions

In this paper, we propose a language-independent rule-based technology that is able to create a phoneticized dictionary. This technology is useable by existing ASR systems and demonstrates significant promise in facilitating the rapid deployment of speech recognition capabilities in newly targeted languages.

Our PLI grammar and IPE interpreter implement a process that enables speech technology to be easily internationalized. The PLI extracts relevant linguistic information by partitioning the process into four discrete steps, thereby reducing computational complexities and greatly simplifying the procedure. The grammar is powerful, simple to use, and effectively encodes the relationship between script and sound.

Simplicity and legibility are the guiding principles of the design because the ultimate goal is to allow non-linguistically trained subjects to create PLI data and thus localize the speech technology efforts. We require support from native speakers with a basic familiarity with computers, but we do not require that these speakers to be linguistic experts. This greatly reduces the cost and effort required to deploy a speech recognitionsystem.

We demonstrated the effectiveness of our approach by comparing experimental error rates with those of an existing English-based speech recognizer. Results show that although our error rates were higher, they can be significantly lowered with minimal expert tuning. We also applied our approach to the Korean language and found that the error rates were comparable to those encountered

with the English-based system.

Currently we are developing the International Phoneticizing Studio (IPS). This tool creates an environment that combines rule resolution mechanisms, a run-time advisory linguistic expert system, a generic speech synthesizer for feedback, and a user friendly graphical user interface. This system will allow native informants to dynamically create a PLI for any given language, using the user's knowledge of a language and using user-friendly tools. As part of our effort, we are extending PLI to more easily handle multiple pronunciations and resolve morphological problems where sentences are not always sequences of discrete words separated by spaces (i.e. Thai, Farci.)

Acknowledgment

This research was supported by the Academic Research fund of Hoseo University in 2006 (20060124)

References

1. H.J.M. Steeneken, and L.F. Lamel, "SQUALE : Speech Recognizer Quality Assessment for Linguistic Engineering", Proceedings ARPA Workshop on Spoken Language Technology, Plainsboro, New Jersey, 1994
2. J.-L. Gauvain and L. Lamel, "Large vocabulary continuous speech recognition: Advances and application," Proc. IEEE, 88 (8) 1181-1200, 2000.
3. H.J.M. Steeneken, and L.F. Lamel, "SQUALE : Speech Recognizer Quality Assessment for Linguistic Engineering", Proceedings ARPA Workshop on Spoken Language Technology, Plainsboro, New Jersey, 1994.
4. T. Matsuoka, K. Ohtsuki, T. Mori, S. Furui and K. Shirai, "Large-Vocabulary Continuous-Speech Recognition Using a Japanese Business Newspaper (NIKKEI)," Proc. Of the ARPA Workshop on Spoken Language Technology, Austin TX, Morgan Kaufmann, Cohen, Ed., 1996.
5. R.I. Damper "Self-learning and connectionist approaches to text-to-phoneme conversion", in Connectionst Models of Memory and Language, Levy J., Bairaktaris J., Bullinaria J., and Cairns P. (eds.), UCL Press, London, 117-144, 1995
6. L. Deng, "Integrated-multilingual Speech Recognition using Universal Features in a functional Speech Production Model," ICASSP '97, 1007-1010, 1997.
7. R., Federking, A., Rudnicky, C., Hogan, Eskenazi, "M. DIPLOMAT," ACL-EACL '97, 1997.
8. T.J. Sejnowski, and C.R., Rosenberg, "Nettalk: a parallel network that learns to read aloud," The Johns Hopkins University Electrical Engineering and Computer Science Technical Report, JHU/EECS-86/01, 1986.
9. M. J. Bert V. Coile, "The DEPES Development System for Text-to-Speech Synthesis," ICASSP '89, 250-253, 1989.
10. S. Hertz, "From text-to-speech with SRS," Journal of the Acoustical Society of America, 1155-1171, 1982.
11. The Unicode Consortium, *The Unicode Standard, version 2.0,* (Addison-Wesley Publishing Company, 1996.

[Profile]

• **Chwa-Cheul Shin**

Chwa-Cheul Shin received the B.S. and M.S. degree in the Dept. of Computer Engineering from Hoseo University, Asan-city, in 1990, 1996. He is Working on his Ph.D. degree in same University. His current interests are in Ubiquitous and speech processing system.

• **In-Sung Woo**

In-sung Woo received the B.S. degree in the Dept. of Computer Engineering from Hoseo University, Asan, in 2000 and M.S. degree from same University in 2003. He is Working on his Ph.D. degree in same University. His current interests are in Ubiquitous and Multimedia processing system.

• **Heung-Soon Kang**

Heung-Soon Kang received the B.S. and M.S. degree in the Dept. of Computer Engineering from Hoseo University, Asan-city, in 1990, 1996. He is Working on his Ph.D. degree in same University. His current interests are in Artificial intelligence and speech analysis.

• **In-Soo Hwang**

In-Soo Hwang received the M.S. degree in the Dept. of Computer Engineering from Hoseo University, Asan, in 1995. He is Working on his Ph.D. degree in same University. He is now a president of ACE Technology co,LED. and His current interests are in Artificial intelligence system.

• **Suk-Dong Kim**

Suk-dong Kim received the B.S., M.S., and Ph.D. degrees in Electronic Engineering from Ajou University in Suwon city in 1982, 1984, and 1993, respectively.
Since 1984, He has been with Dept. of Computer Engineering, Hoseo University, Asan. He is now a professor and his current research interests include speech recognition, Ubiquitous and speech analysis. He is a member of the Acoustic Society of Korea.