Internationalizing Speech Technology through Language Independent Lexical Acquisition

Bertrand A. Damiba
Alexander I. Rudnicky
{bd1r,air}@cs.cmu.edu

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

Software internationalization, the process of making software easier to localize for specific languages, has deep implications when applied to speech technology, where the goal of the task lies in the very essence of the particular language.

A great deal of work and fine-tuning normally goes into the development of speech software for a single language, say English. This tuning complicates a port to different languages. The inherent identity of a language manifests itself in its lexicon, where its character set, phoneme set, pronunciation rules are revealed.

We propose a decomposition of the lexicon building process, into four discrete and sequential steps:

(a) Transliteration code points from Unicode.

(b) Orthographic standardization rules.

(c) Application of grapheme to phoneme rules.

(d) Application of phonological rules.

In following these steps one should gain accessibility to most of the existing speech/language processing tools, thereby internationalizing one's speech technology. In addition, adhering to this decomposition should allow for a reduction of rule conflicts that often plague the phoneticizing process.

Our work makes two main contributions: it proposes a systematic procedure for the internationalization of automatic speech recognition (ASR) systems. It also proposes a particular decomposition of the phoneticization process that facilitates internationalization by non-expert informants.

1. INTRODUCTION

Many interesting questions arise when adapting existing speech systems to languages other than the original target language [12]. Most of the assumptions that have found their way into the core of single language designs do not necessarily hold when applied to other languages. For they are often expressed with different character sets, have different phoneme sets and pronunciation rules along with other specificities. Moreover, native speakers of the language would have tuned performance over time.

A number of approaches have been proposed. Some advocate rebuilding from scratch new systems adeptly suited for a target language [9,8]. Others prefer rebuilding new statistically based systems aimed at cross language portability [4]. Although valuable, these approaches can be very costly and result in redundant work. When dealing with

rapid deployment systems, an approach that would make the most use of existing systems and require smaller scale commitment would be, perhaps, better suited.

We have been exploring problems of automatic speech recognition and text-to-speech synthesis portability in the context of the DIPLOMAT project [5], successfully dealing with Serbo-Croatian, Haitian Creole and Korean languages. The goal of DIPLOMAT is to create tools for rapid deployment.

In speech technology terms, a language mostly finds its uniqueness in the way it sounds and in its script, both of which are specified in its lexicon. The lexicon is the most localized part of any speech system, since, once the character set issue is solved many of the other components of a system need no further internationalization.

Some other approaches strongly rely on machine learning [11], and are therefore dependent on the amount and quality of existing data, an assumption that doesn't hold for many languages. Our approach relies on the availability (or tele-availability) of a native informant and their effective use of their knowledge of the language. We do not however assume that this needs to be someone with formal training in linguistics or speech recognition, only that they possess a basic familiarity with computers.

User friendliness is, therefore, an important factor in any realistic attempt at making systems multilingual. A simple design will allow a user-friendly environment, therefore opening the process to non-linguistically trained native speakers. Of the existing approaches to language independent phoneticizing grammars [6,17], most do not consistently address the character set issue, neither do they offer grammars that are legible by non-linguistically trained experts.

This paper proposes an extended, language independent phoneticizing process, consisting of four steps.

(a) Transliterating code points from Unicode.

(b) Phonetically standardizing rules.

(c) Implementing grapheme to phoneme rules.

(d) Implementing phonological processes.

The application of these four steps will tranform a Unicode string into its corresponding phonetic string, solving the character set issues along the way. We will also present a simple grammar that specifies these steps: the PLI (Phonetic Language Identity) format. Also discussed in this paper are the first implementation of a PLI interpreter, the IPE (International Phoneticizing Engine), and its use and results when applied to Korean using Carnegie Mellon's Sphinx III speech recognition system [7,10].

This paper also addresses the reduction in complexity through the decomposition of the phoneticizing process allowed by a sequential rather than global rule application.

2. THE FOUR STEP PHONETICIZING

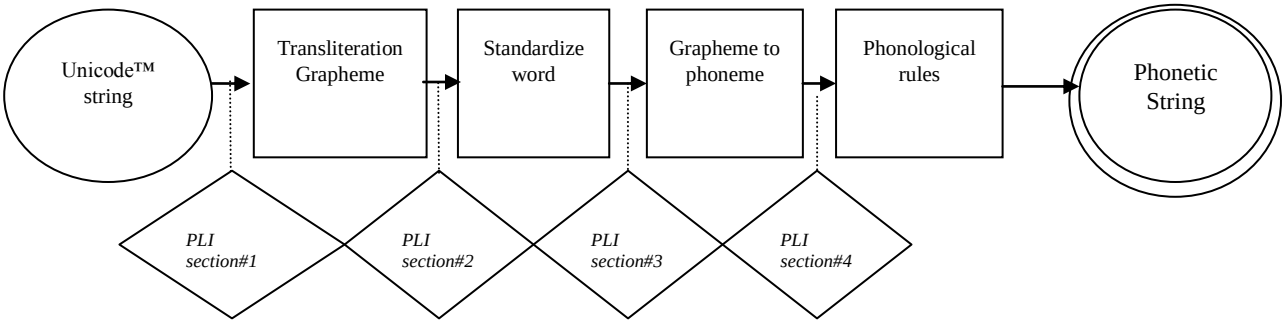


Figure 1. Language Independent Phoneticization in Four Steps

The basic scheme of the approach is shown in Figure 1. The four steps consist of a Unicode transliteration followed by a normalization of the orthography, a phoneticization of the normalized string and finally a phonological pass. Each discrete step has a well-defined goal, which is simple enough to potentially open up the process to non-linguistically trained users. The transformation process is rule driven and involves four steps of rules (PLI sections). The work of phoneticizing a language consists of creating these four rule sets. Unlike machine learning-based approaches [3], our ultimate aim is not to completely automate the lexical acquisition process, but rather to structure it in a way that will allow native speakers (not necessarily a linguist but computer literate) to make speech technology multilingual. The output of the process is a sequence of phonemes expressing the pronunciation of that Unicode string. Below, we will take a closer look at each of these steps.

Undesirable rule interaction is often the single obstacle in the successful rule-based phoneticizing of a language [17]. By dividing the rule space in three (PLI sections 2-4), the user only needs to ensure that the rules created are consistent within the space they address. We believe that this approach drastically reduces the complexity of the task.

Transliterating from Unicode to ASCII

As Unicode [16] has become the commonly accepted standard universal character set, it opens our process to most known languages. Unicode is a fixed size character set, where each text element is encoded in 16 bits (UCS-2); this allows uniformity across languages. More importantly, the Unicode consortium [16] has set standards for the processing of many scripts that defy the assumptions made by ASCII (i.e. bi-directional algorithms, Hangul syllable decomposition/composition algorithm etc...) and can be helpful to speech technology. For these reasons, Unicode must be included in any attempt at language independent lexical acquisition.

The process of transliterating from Unicode has the goal of mapping each relevant Unicode code point used in the target language to an ASCII string to be used in the later

steps. This process defines a Unicode code space for that language and creates an ASCII mapping that can be used to refer to a particular text element.

Transliterating allows for extracting all the information contained by the text element. ASCII, designed for American English assigns a code point to each letter of the Latin alphabet. In some language one text element encodes more than one linguistic phenomenon. For example, in French vowels often carry diacritical marks, in Hangul where each text element is a syllable, a text element may carry up to four jamos (that is, single letter of the Korean script). Transliteration allows us to create our own, string-based internal character, well suited for phonetic processing which has the virtue of fitting with existing ASCII based ASR systems.

Speech technologies often exploit the relationship between the spoken word and its written representation, yet not all languages have a phonetic script (Mandarin, Cantonese). Transliterating allows us to recreate the script at the text element level, and recreate, for the purposes of the task at hand, that crucial relationship between the written word and the spoken word.

This illustrates that the complexity of the transliterating step varies across languages, a trivial step for phonetic languages with few characters, a more involved step for languages with extensive ideographic scripts.

Standardizing the orthography

Languages carry in their orthography a certain complexity as a result of their history. Often the orthography to sound relationship is not quite intuitive (i.e. English: "knight" sounds more like "nite", in French "paon" sounds more like "pan", in Korean: f같이 sounds more like 가치g). Other languages are quite flexible in the way they are written, allowing several orthographies for the same word (in Haitian Creole "pwezidan" and "presidan"). In the case of homophones, the orthography marks a semantic difference (i.e. English "know", "no"). This creates the need for a phonetic standardization of sorts for the purposes of speech technology, where often these artifacts are obstacles to that important script to sound relationship.

This step also gets us closer to a context independent pronunciation of subword units, alleviating the load of the subsequent phoneticization steps [6].

Here again, because the goal of this process is intuitive and self-explanatory, a non-linguistically trained native speaker could perform it.

From Graphemes to phonemes

With a standardized orthography, this step is meant to implement basic grapheme to phoneme mappings. All the remaining context dependent pronunciation combination ought to be addressed during this step. Phoneme interaction such as nasalization need not be created here in order to reduce collisions.

Phonological processes

In any given language, regardless of the orthography some pronunciation rules are solely based on sound. In French, when some identical plosives are repeated, only one is pronounced (i.e. "toure**tt**e": `T UW R EH T T` → `T UW R EH T`). This section is also meant for differentiating between allophones, depending on their phonetic context.

Logically stemming from this approach is a grammar that implements it while keeping with the theme of simplicity and legibility. It explicitly implements our assertion that the phoneticization process can be effectively modeled as a sequence of locally simple transformations. Its purpose is to embody all the information about a language that is relevant for speech technology (character set, phoneme set, grapheme to phoneme relationship etc...), thus the name : phonetic language identity.

3. THE PLI FORMAT

The overall format of the PLI is a text file very much similar to a mapping table of the form:

```
Source_substring [tabulation mark] Target_substring
```

The PLI rule set is divided in four subsections representing each a discrete step in the transition from Unicode™ to a phonetic string. Each section is separated by keywords "#1", "#2", "#3" and "#4" on a line by themselves. All sections need to be present and in order.

The sections (see Figure 1)

Section 1: A Unicode code point in hexadecimal uppercase followed by a tabulation mark and the transliteration into ASCII. Referring to the code point explicitly (instead of the 16 bit text element it addresses) allows the PLI format to be in ASCII (thus allowing electronic transmission without encoding), yet making it able to refer to Unicode. It is recommended to add a comment containing the actual text element after each entry in PLI section #1 line, in order to allow some degree of verification. Putting it after a comment mark will not compromise the assumption that the PLI format is ASCII compliant when processing it, because the comment marker will make the grapheme invisible to the interpreter.

```
D4AD        {pVt} //팥
```

Section 2: Transliteration [tab] standardized transliteration. That is:

```
kn          n
```

Section 3: Normalized transliteration [tab] Phonemes. That is:

```
th          [TH]
i<vowel>    [AY]<vowel-sound>
```

Section 4: Phonemes sequence I [tab] Phoneme sequence II. That is:

```
[G][G]      [G]
[X]         [K][S]
```

The comments

All the strings followed by the keyword "//" ought to be ignored for they are meant for comments when one edits the PLI manually. That is:

```
            //The following rules take care of...
            th      [TH]   //This   rule   phoneticizes   "th"
sound
```

The space marker

The keyword "+" is meant to represent a space so inter-word phonetic interaction can also be expressed. This allows the PLI to be both an exception dictionary and a rule based grammar system.(see Figure 2)

```
s+<vowel>  [Z]+<vowel-sound>
```

The Null phoneme

There is a special case phoneme, "#" which will not be printed. This phoneme can conveniently be used for the unpronounced grapheme sequences:

```
[HH]+       #+
```

The grouping variables

The goal in using grouping variables is to simulate the behavior of a class of units, without having to explicitly list them all. When the PLI is processed, these variables will be expanded to their enumerated sets. The order of the enumerated units is very important because each unit is matched by its position when enumerated.

One can set a grouping variable by entering:

```
<variable_name> = {unit1 unit2 unit3 unit4 ...}
```

One can use a grouping variable by entering:

```
ph<variable_name1>  f<variable_name2>
```

```
//Grouping Variables
<vowels> = {a e i o u}
//groups the vowels in one variable
<vowel-sound> = {[AA] [EH] [IH] [AO] [UW]}
//groups the vowel sounds in one variable
i<vowels>  [IY]<vowel-sound>
//using the grouping variable to simulate the sound of "i"

//Space Marker
s+<vowel>  [Z]+<vowel-sound>
//implements the French inter-word "liaison"
+colonel+  +kernel+ //PLI can be used as an exception dictionary
```

Table 1. Grouping Variable and Space Marker Syntax in PLI

The declaration of a grouping variable needs to precede its use (see Table 1). As well, the individual elements in the two vectors (<variable_name1> and <variable_name2>) need to have a one to one mapping between them, as the translation is implicit.

The only exception to this rule is in case the target doesn't contain any variables, in which case the entire expanded source will map to the same target:

```
o<variable_name1>    o#
        or
o<variable_name1>    o
```

When creating a PLI file, 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 use of the different sections should freely be adapted if the need arises. For example, in developing a PLI for English, phonemes were used in PLI section #2 due to the lack of context independent graphemic representation of some phonemes.

The use of the grouping variable, since they are user-defined, allows all levels of sophistication. A non-linguist might just group consonants and vowels together, while a linguist could create grouping variables for as many phonetic features as desired. The flexibility of this grammar combined with the legible "source *tab* target" format, make for a powerful language for linguist and non-linguist alike, such a grammar requires an interpreter build around the same goals and assumptions.

3. THE IPE: INTERNATIONAL PHONETICIZING ENGINE

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. It adheres closely to the format pointing out any inconsistencies of the PLI it interprets as it encounters them. The user can specify the output format, in other words, one can choose which output of each of the four steps to print. Therefore if only the transliteration is needed, then only the transliteration of the Unicode text file will be output.

The IPE also offers a “trace” mode (see Table 2), which allows the user to follow how each of the four steps affect the Unicode string (by displaying the rule and its line number in the PLI), allowing for the debugging of the PLI rules.

At each section all the rules are sorted (descending order) by the length of the source string. Rules with equal source string length the order given will be kept. This permits the

more drastic rules to first take effect, and other rules to follow. The rules are applied in a strict left to right manner, although the order of the substrings they affect will be decided by the aforementionned rule ordering. Having defined our process, specified a grammar and described its interpreter, we evaluated some of our assumptions and theories.

```
-------------------------------------------
#TRANSLITERATION FROM UNICODE
  internationalize
#TEXT TO STANDARDIZED TEXT
  internashahnal[AY]z
    477 ize+    ->  [AY]z+
    558 tion    ->  shahn
#TEXT TO PHONEMES
[IH][N][T][ER][N][AH][SH][AH][N][AH][L][AY][Z]
    689 sh      ->  [SH]
    693 er      ->  [ER]
    704 ah      ->  [AH]
    712 l       ->  [L]
    712 n       ->  [N]
    712 t       ->  [T]
    712 z       ->  [Z]
    713 a       ->  [AH]
    713 i       ->  [IH]
#SOUND TO SOUND
[IH][N][T][ER][N][AH][SH][AX][N][AH][L][AY][Z]
    799 [SH][AH][N]     ->  [SH][AX][N]
-------------------------------------------
```

Table 2. Output of the IPE in Trace Mode (phoneticizing the English word *internationalize*).

4. EVALUATION

I Creating PLIs for Korean, Haitian Creole, English and Serbo-Croatian

To understand the practical relevance of the process we have defined, we created PLI rule sets for 4 different languages. In this section we present the results of this study and characterize the magnitude of the effort involved. The times cited reflect the amount of editing effort necessary to create the relevant PLI rules; they do not reflect prepatory work (such as creating word lists etc...).

In order to create PLIs the most frequent words of our text corpus were chosen. The assumptions were that, most pronunciation phenomena would be encountered in that subset and secondly, the coverage that these high frequency words have on the language would result in a better performing PLI when applied to natural text.

a) *Haitian Creole*

(See Appendix B)

Created by one native French speaker with the collaboration from a fluent Creole speaker.

Haitian Creole is a language for which only recently have standardized orthographic representations been proposed. On the one hand the relationship between the sound of the language and the script is quite simple, on the other hand one can find many orthographic representations for the same word. For our experiment, 600 words were used to build the PLI.

PLI section #1: (100 rules, 20 minutes to create)

Imported from Unicode all the basic alphabetical characters found in ASCII in a case insensitive fashion, in addition to the ones that have diacritical marks, which were made more explicit (i.e ò → o`).

PLI section #2: (7 rules, 5 minutes to create)

These rules tend to standardize the Haitian Creole orthography, since most words are written very much like the way they sound, little else is needed. It might be noted that Haitian Creole has many borrowed words in its lexicon that defy its own pronunciation rules. These lone words should be entered as exceptions in this section. Our text corpus, obtained through the translation by 3 native speakers, didn't offer the variety of orthographies that can be found in the language at large; for which more standardization rule would probably be required.

PLI section #3: (45 rules, 5 minutes to create)

A simple one to one mapping of graphemes to phonemes implemented here.

PLI section #4: (5 rules, minimal time to create)

Simple nasalization exceptions were inserted in this section.

b) Korean (Hangul)

(See Appendix C)

Created by 2 native speakers of Hangul. Korean presents an interesting challenge because it has a very large and very different character set from English, yet is a phonetic script requiring further rules. A total of 900 words were used in building the PLI.

PLI section #1: (11,179 rules, 10 minutes to create):

Each Unicode Hangul phonetic Syllable was mapped to a corresponding string of, usually, three jamos enclosed in curly brackets. The 11,179 code points were mapped automatically with the use of a very simple script.

PLI section #2: (240 rules, 8 hours to create)

Hangul is a regular language, where each jamo has a pronunciation depending on its position in the 3-jamo wide syllable. There are some inter-syllable interactions that affect the pronunciation; these rules were expressed here. In particular, some composed jamo (where one jamo represents the ligature of two other) have pronunciations entirely dependent on their context. The use of encasing marks in PLI section #1 was quite useful in expressing these types of rules (e.g.`M}{X → r}{m`).

PLI section #3: (1 rule)

Implements the three possible pronunciations depending on the position of the jamo.

PLI section #4: (10 rules, 10 minutes to create)

Implements some nasalition and deletion rules

c) English

(See Appendix D)

Created by a native speaker and a fluent non-native speaker.

English (expressed in ASCII) offers the easiest language to import from Unicode. However English doesn't have a very good sound to script relationship. A total of 1030 words were used in building the PLI.

PLI section #1: (54 rules, minimal time)

PLI section #2: (220 word exceptions, 286 standardizing rules: 506 total rules, 25 man-hours to create):

Standardizing a language that contains so many exceptions to its standard pronunciation rules was a true challenge. This PLI was built with only the IPE to check results. This illustrated the need for a rule collision detection mechanism (see International Phoneticizing Studio). Substring exceptions (i.e `sure+ → zher+`) were, by design, easily taken care of. In the cases where some sounds weren't expressible in a context independent way, phonemes were used (e.g. `tial+ → sh[AH]l` , *see next section*).

PLI section #3: (47 rules, 45 minutes to create)

Due to the large amount of standardization, we were able to drastically reduce the number of grapheme to phoneme rules.

PLI section #4: (63 rules, 1 hour to create)

Some of the "-ed" ending rules were inserted here, as well as many deletion rules (<consonant-sound>[HH] → <consonant-sound>)

d) Serbo-Croatian

(See Appendix E)

Created by a linguist (non-Serbian speaker) with the help of linguistic reference manual on Serbo-Croatian. Like Haitian Creole, Serbo-Croatian is written very much like the way it sounds.

PLI section #1: (51 rules, 10 minutes to create)

Serbo-Croatian has a pretty loose character set, some graphemes can be expressed by the sequential combination of other graphemes or simply one text element representing their ligature (e.g. one text element: "nj" and two text elements: "n" followed by "j"). The section was made to encompass most cases, in order to allow some flexibility in the way the text corpus was written.

PLI section #2: (1 word exception)

PLI section #3: (30 rules)

Simple grapheme to sound mapping.

PLI section #4: (0 rule)

e) Discussion of experiences with the different languages

After building PLI files for these very different languages, some lessons and insights were learned.

PLI section #1 should be by nature expansive; whether it's importing some ancient syllables for Hangul, or deal with all types of combination people might use to represent ligatures in Serbo-Croatian. Because PLI section #1 defines the code space of the Unicode text corpus, flexibility is key. It is probably the most important section for scripts that defy many of the assumptions made by ASCII (i.e. Hangul), because the way the code points are imported will affect the complexity of the following sections.

PLI section #2 serves many purposes, the most important one is to remove ambiguities in the sound to script relationship of a particular language. It presented itself as a real challenge for English, a language rich in exceptions with a weak script to sound relationship. Standardizing the orthography is the most laborious task of extracting phonemes from words. Because of the large number of rules in this section, some rule collision occurred despite the division of the rule space in three distinct subspaces. In languages with a good script to sound relationship (Haitian Creole, Serbo-Croatian), the PLI section #2 was used to respell foreign words. Or, in the case of Haitian Creole to standardize a language that allows multiple orthographies for the same word in order to allow corpora with the same orthographic flexibility the language permits.

In PLI section #3 for all languages mapped graphemes or sequences of graphemes to a phoneme. Across languages it was the simplest section to create for it is based on the assumption that, after PLI section #2, the language's script is standardized.

PLI section #4 across languages dealt with phonemic phenomena (stress, nasalization etc...), but turned out to be more a "clean-up" section were improbable phoneme sequences were reassessed (e.g. `<consonant>[HH]<consonant>` in English).

Pli section # / Languages	**Pli section #1** Unicode import	**Pli section #2** Standardizing orthography	**Pli section #3** Grapheme to phoneme	**Pli section #4** Phonological processes
English	54	506	47	63
Korean	11,769	240	1	10
Haitian Creole	100	7	45	5
Serbo-Croatian	51	1	30	0

Table 3. Summary of Number of Rules Needed for each PLI Section across Languages

f) The use of phonemes in PLI section #2 of English

Although PLI section #2 is advised to be used for rewriting a word in an orthography that is more phonetically correct, while developing a PLI for English, we used phonemes in that same section (of the 286 standardizing rules, 47.5% contains phonemes). In producing this section, one is to express some sounds using graphemes. While executing this standardization some rules may affect the boundaries of these graphemic representations of phonemes. Sometimes these effects are desirable, other times not. The use of phonemes in the English PLI section #2 is to address the occurrences when these effects are not desirable in other words, block any further rule application when no more standardization is required.

Desirable side-effect at boundaries:

Consider 2 rules:

```
tion      shan  //Rule #1
ns+       nz+   //Rule #2
```

consider the word "portions"

after rule #1 becomes:

```
porshans
```

after rule #2 becomes:

```
porshanz
```

"porshanz" is indeed a decent graphemic approximation of "portions" which phoneticizes to "[P][AO][R][SH][AH][N][Z]".

Undesirable side-effect at boundaries:

Consider the following words "majority" and "write". The last phoneme of "majority" is [IY], one could enter at PLI section #2 the following rule:

```
ty+  tee+      //Rule #3
```

After Rule #3 our two words become:

```
majoritee
write
```

One could rightfully consider the following rule to implement the pronunciation of words containing the substring "rite"

```
rite rayt      //Rule #4
```

After Rule #4 our two words become:

```
majorayte
wrayt
```

As shown above, at a certain level of standardizing, some pronunciations become final and require no further rule applications. More rules can be devised to get around this problem (i.e. using uppercase characters or subword segmentation for rule blocking), but we believe that, as a time saving solution, that using phonemes, which will finalize the subword pronunciations, is a valid approach.

```
ty+  t[IY]+
ite  ayt
```

These words become:

```
majorit[IY]
wrayt
```

Other valid approaches may include adding the words as exceptions in such cases. But we believe that 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. Another approach uses a prior syllabification step to alleviate this very problem, thus limiting undesirable application of rules at these boundaries [6]. We believe that because the consistent spotting of syllables can be illusive to non-linguistically trained users, this step didn't fit in a system ultimately meant to open the phoneticization process to non-linguists.

This remedy was not needed in other languages. This emphasizes that the guidelines of the PLI grammar ought to be flexible enough to allow a degree of adaptation for it to truly be internationalized. Due to this flexibility, the four step of conversion process gives the user enough choice and adaptability to allow the application of the proposed phoneticizing process on a wide variety of languages.

g) Does seperating the rule space in three discrete subspaces streamline the process?

We claim that separating the rule space in three distinct and sequential zones (not counting PLI section #1: the transcoding phase) will structure the process in such a way that will allow non-experts to provide basic phonetic knowledge of a language by defining three sequential intuitive goals. In addition, a desirable effect of this partitioning is that it prevents the goal specific rules to interact with each other.

By specifying intuitive goals, we reduce a task that would require a reasonable knowledge of phonetics to a series of tasks that are specific and simple.

> PLI section #2: Regularizing a grammar is a task that is quite intuitive, spelling words according to the way they sound is often a phase we all go through while learning a new language. So we all have a strong familiarity with it. On the other hand, some decisions may be quite arbitrary and introduce idyosynchrasies to the PLI that may complicate collaborative efforts. But since the PLI is a rapid deployment resource, fine-tuning and collaboration are not the immediate goals here.
>
> PLI section #3: Phoneticizing a regular language is a simple task where limited context is needed to infer a general rule for the pronunciation of a particular letter group. (*see PLI #3 for Haitian-Creole and Serbo-Croatian*)
>
> PLI section #4: Dealing with phonological processes can also be an accessible granted the previous sections were well implemented.

We also believe that this partitioning scheme also allows us to reduce the number of rules and limit the number of "patch-up" rules. Since each section makes strong assumptions relating to the state of the input, the task of achieving its goal is limited.

PLI section #2, we assume that the input is an ASCII string, we went from the UCS-2 plane to the ASCII plane in PLI section #1. We ignore all the character set specific issues.

PLI section #3, we assume that the input is an orthographically regularized string, we therefore ignore all sorts of orthographic phenomena as we phoneticize.

PLI section #4, we assume that the input is a phonemic string, with no graphemes. We can ignore all the grapheme to phoneme rules.

In order to prove the aforementioned claims, we decided to create two separate English PLIs. One that would use the proposed three level rule set, another one in which all the rules would be expressed in the same section. In both tasks the goal was to achieve acceptable phoneticizations for the 1030 most used words in WSJ data. We will compare the results using several criteria: complexity of the task (assessed in man-hours), number of rules, number of exceptions.

	Complexity of task (man hours)	**Total number of rules**	**Total number of exception words**
English all 3 sections	<30	620	220
English with a single section	>60	819	326

Table 4. Comparing the PLI using one transformation section, with the one using all three

As the results show the complexity of the task rises significantly when local goals aren't defined. We can also group the increase in number of exception words to the need to "patch-up" undesirable affects of some rules.

II Evaluating the English PLI

While the proposed approach provides a useful process for phoneticizing a written language, there is no inherent check for the accuracy of the lexicon produced. It is however possible to assess the accuracy of a PLI-based lexicon by comparing it to a handcrafted version of the same lexicon. Such a comparison was possible to make for the English PLI, by comparing it with a well-established pronunciation dictionary of American English developped for use in speech technology applications, the CMU dictionary [15].

Context independent phonetic acoustic models 2997 word dictionary LM perplexity 52.77 68 speakers 136 utterances ATIS 93 Task	**Word Error Rates (%)**
CMU Dictionary 0.5b	12.11
IPE English:	30.72
IPE English with minimal changes*:	20.07

Table 5. Error Rates using English PLI with SPHINX III ASR System Vs. CMU dictionary 0.5b

*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.)

When comparing the CMU dictionary (see Table 5) into which a great deal of expert work has gone in for several years, with the output of our approach when applied to English, two lessons can be learned. First with no additional work, the scheme allowed an initial error rate of 30.72%, which considering the non-cost of the approach is a very good start for rapid deployment system. The second lesson is the significant decrease in error rate with very minimal expert fine-tuning (34.3% relative decrease of the word error rate). Thus proving that this approach drastically minimizes the deployment time and cost once the PLI is created.

III Evaluating the Korean PLI

a) SPHINX III ASR system with a Korean PLI

As shown in Appendix A, the approach proposed in this paper was applied for Korean, using the SPHINX III ASR system [7,10] combined with the CMU-Cambridge Language modeling Toolkit [1]. Since the SPHINX programs and the CMU-Cambridge Language toolkit are not Unicode compliant, the output of the PLI section #1 was used for the word references in the pronunciation dictionary, as well as the text corpus for the language model.

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. The utterances were phoneticized, we then 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 (13.11 minutes of speech) uttered by speakers not used in the acoustic training.

LM text corpus size: 14358 Unique words in transcript: 310 Dictionary size: 8550 words	**Trigram Language Model** Perplexity: 6.25 Entropy: 2.64 bits	**Bigram Language Model** Perplexity: 38.48 Entropy: 8.40 bits	**Unigram Language Model** 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

Table 6. Error Rates using Korean PLI with SPHINX III ASR system

*In written Korean, morpheme boundaries are looser than in English (i.e the word "국민교육헌창" can also be correctly written as the sequence of the words: "국민", "교육" and "헌창"), therefore word error rates for Korean systems can be pessimistic. As an informative counter balance, syllable error rates are given here as an optimistic assessment of the recognition results.

b) A Cross-Language Comparison: Sphinx III English vs. Sphinx III Korean

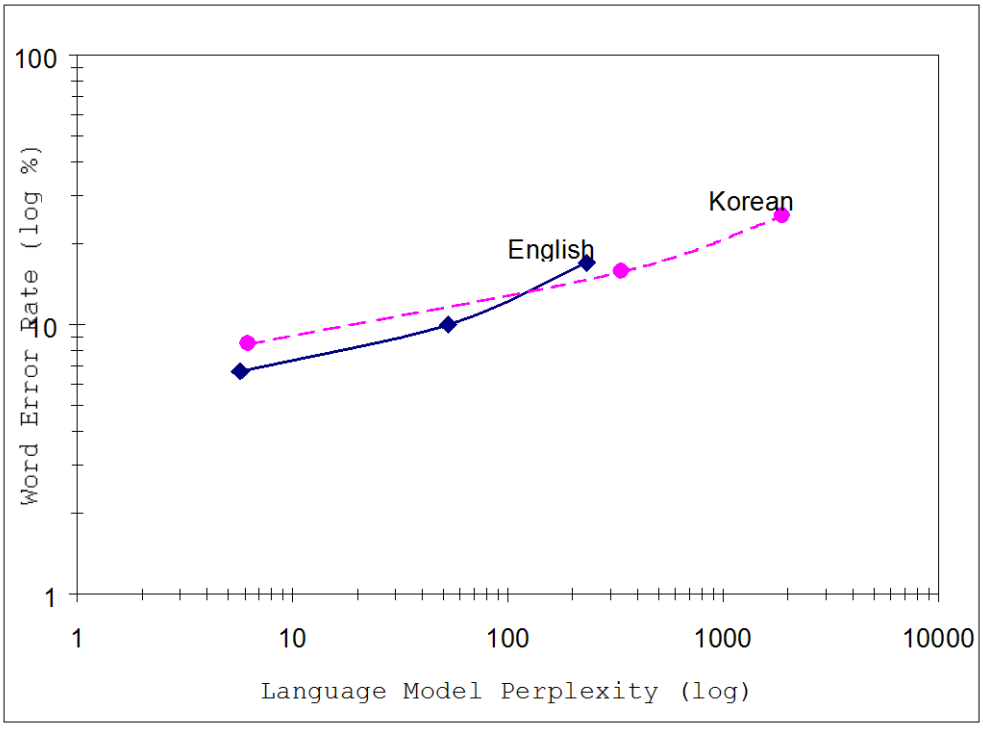


Figure 2. Affect of the Language Model Perplexity on the Word Error rates of Korean and English ASR Systems

Cross-language comparisons of ASR systems can be fickle because the factors to consider are innumerable. As an attempt to compare the robustness of our newly created Korean context dependent acoustic models, with that of our established English context dependent acoustic models (ATIS recognition task), we chose the perplexity of the language model as the decaying factor. As we can see in Figure 2, the overall decay pattern of the word error rates of English and Korean is similar. We can however note that for perplexities below 100 the English acoustic models fare better. But beyond perplexities of 100, the Korean acoustic models seem to do even better than the English models. In our pronunciation dictionaries, Korean words tended to be phonetically longer than English words (we observed an average of 8.31 phonemes per word in our Korean dictionary and only 6.24 phonemes per word in our English dictionary). We can deduce that the reduced phonetic confusability of the Korean lexicon might be the reason for the slower increase in word error rate at high language model perplexities.

Most importantly these results show that the Korean acoustic models, even when put under challenging conditions behave in a similar way to the English acoustic models. Since the Korean acoustic models were built using exclusively a pronunciation dictionary automatically created by the Korean PLI and the English acoustic models were conventionally trained with a handcrafted dictionary, we believe that this validates the proposed phoneticization process as a viable alternative to handcrafted lexicons.(trading optimal quality for a rapid deployment time)

5. FUTURE GOALS

The PLI grammar and its interpreter set forth a process that enables speech technology to be internationalized. Simplicity and legibility were the guiding principles of their design because the ultimate goal is to also allow non-linguistically trained subjects to create PLIs and thus localizing the speech technology at hand. Currently in development is the International Phoneticizing Studio (IPS), an environment that would allow these native informants to dynamically create a PLI for any given language. With a combination of rule resolution mechanisms, a run-time advisory linguistic expert system, a user friendly graphical user interface and a generic speech synthesizer for feedback, the user's knowledge of the sound to script relationship of a given would be extracted and stored in a PLI.

There is also a need to allow in the PLI grammar a way to allow multiple pronunciations, for they are often used in speech technology, while not compromising the simplicity and the legibility of the grammar.

In addition many languages have their own morphological identities, in which the assumption that a sentence is a sequence of discrete words separated by spaces doesn't hold (Thai, in some instances German and Korean). This illustrates the need for a Unicode based International Morphological Tagger that would extract morphemes based on a base dictionary with features.

6. CONCLUSION

We hope this paper opens discussions on many fronts.

We consider the following to be key:

- •Introduction of a uniform scheme for spoken/written language integration based on the Unicode standard. Which is much more than a super character set; it offers a variety of useful algorithms and standards for the processing of many scripts.
- •Introducing software internationalization concepts into speech technology.
- •Lowering the expertise level required for lexical acquisition to language fluency.
- •Establishing a process for low cost rapid deployment systems.
- •Explore an alternate approach for rule based phoneticization based on factoring the rule space into normalization and phoneticization components.
- •Defining a formal grammar for expressing mappings required for the processing of a particular language (character set, phoneme set, character-to-phoneme relationship and pronunciation exceptions).

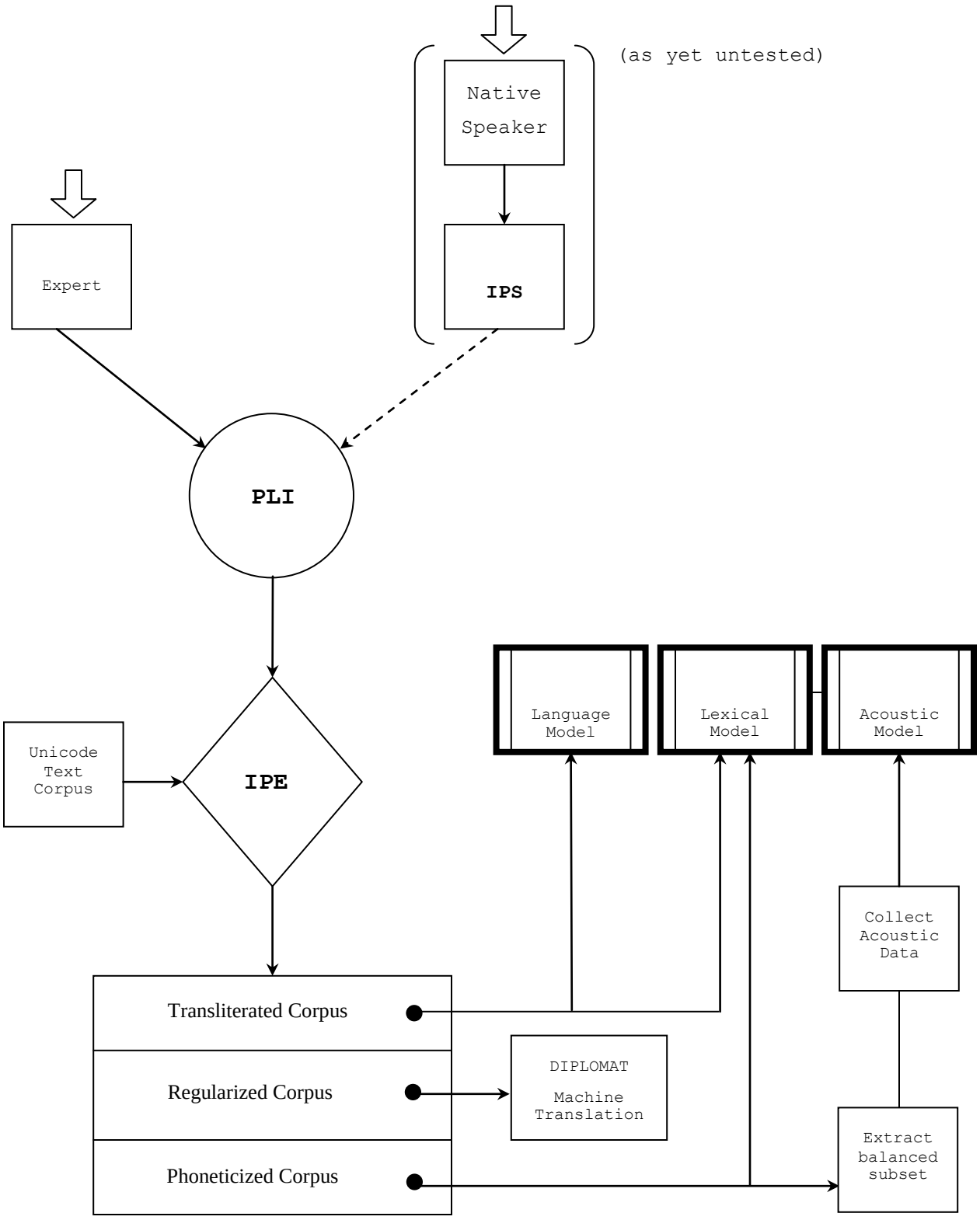
We evaluated our approach and obtained several concrete results. We established working PLIs for English, Haitian Creole, Korean and Serbo-Croatian. We established that the increased error rate our automatically generated dictionary incurred (when compared with a hancrafted dictionary) was easily reduced with well-defined fine-tuning. We also applied the proposed internationalization procedure to the SPHINX III ASR combined with the CMU-Cambridge Toolkit, and we obtained a Korean ASR system with a good recognition rate and acoustic models with a robustness comparable to the English acoustic models. We also proved that, in the phoneticizing process, dividing the rule space helped in reducing the complexity of lexical acquisition.

Several questions remain. Can the ultimate goal of establishing an interactive tool to dynamically create a PLI file be realized? Can the PLI grammar include ways to encode multiple pronunciations without jeopardizing the legibility of the grammar? Since, currently, a PLI file will provide an initial gain, but all subsequent lexical refinements are not stored in the PLI. Although we were able to test our procedure on several challenging languages, applying it to more languages will emphasize the places where improvement of the procedure is needed. Neither does this paper address the issues of data acquisition (acoustic and text) often required when adapting a system to a new language.

Finally we believe that this paper's main contribution is to attempt to combine the established fields of speech technology and linguistics with the relatively new fields of software internationalization and human computer interaction to try to answer the question: "Can we make established speech technology multilingual at a low cost?"

7. ACKNOWLEDGEMENTS

We would like to acknowledge the valuable advice and contributions from Joseph Kenney, Dr. Suk-Dong Kim, Scott Hansma, Eric H. Thayer, Dr. Mosur Ravishankar, Dr. Roni Rosenfeld and Photina Jang.



Appendix A: Internationalizing SPHINX through language independent lexical acquisition

```
//**********************Haitian Creole PLI*****************************
<vowel> = { a e i o u y a` e` e' o` o' }

#1
//    Implements case insensitivity and transliterates
//    characters with diacritics.
0041  a   //A
0042  b   //B
0043  c   //C
[...]
0061  a   //a
0062  b   //b
0063  c   //c
[...]
00C0  a`  //À     Code points with diacritics
00C1  a'  //Á
00C2  a^  //Â
00C3  a~  //Ã
00C4  a:  //Ä
00C7  s   //Ç     C with cedilla imported as "s"
00C8  e`  //È
00E7  s   //ç     C with cedilla imported as "s"
00E8  e`  //è
[...]

#2
//    Stardardizes the few inconsistancies of Haitian Creole
x           ks
qu          k
q           k
ph          f
pw          pr
<vowel>-n   <vowel>nn // Standardizes the use of "-"

#3
//    Since Creole is mostly written the way it sounds
//    most of the rules below are just grapheme->sound mappings
+w    +[W]
ui    [U][IY]
b     [B]
d     [D]
g     [G]
[...]

#4
//    Deals with fake nasalization Sound to Sound Rule.
[EN][N]     [EH][N]
[AN][N]     [AA][N]
[ON][N]     [AO][N]
[UN][N]     [AO][N]
[IN][N]     [IY][N]
[...]
```

Appendix B: Haitian Creole PLI (sample)

```
//************************** Korean PLI ****************************
//    Some Grouping Variables declarations
//    The List of all "Jamo"s
<jamo> = { a A c j e E U h i N k g x m n l L o O p b q r H G M z B Z D
s C K P S T t d u w W v V J y Y f F I Q R X _ }

//    The List of the phonemes of these JAMOS
//    when in initial position of the syllable
<initial-sound> = { [AA] [AE] [CH] [JH] [EH] [AO] [EU] [HH] [IY] # [K]
[G] [K] [M] [N] [N] [N] [OW] [W][EH] [P] [B] [P] [R] [*] [*] [*] [*]
[*] [*] [*] [S] [JJ] [KK] [PP] [SS] [TT] [T] [D] [UW] [W][AA] [W][AE]
[W][EH] [W][AO] [W][IY] [Y][AA] [Y][AE] [Y][EH] [Y][AO] [EU][IY]
[Y][OW] [Y][UW] # # }

//    The List of the phonemes of these JAMOS
//    when in medial position of the syllable
<medial-sound> = { [AA] [AE] [CH] [JH] [EH] [AO] [EU] [HH] [IY] # [K]
[G] [K] [M] [N] [N] [N] [OW] [W][EH] [P] [B] [P] [R] [*] [*] [*] [*]
[*] [*] [*] [SS] [JJ] [KK] [PP] [SS] [TT] [T] [D] [UW] [W][AA] [W][AE]
[W][EH] [W][AO] [W][IY] [Y][AA] [Y][AE] [Y][EH] [Y][AO] [EU][IY]
[Y][OW] [Y][UW] # # }

//    The List of the phonemes of these JAMOS
//    when in final position of the syllable
<final-sound> = { [AA] [AE] [TC] [TC] [EH] [AO] [EU] [TC] [IY] [NG]
[KC] [KC] [S] [M] [N] [CH] [HH] [OW] [W][EH] [PC] [PC] [S] [L] [*] [*]
[*] [*] [*] [*] [*] [TC] # [KC] # [TC] # [TC] [TC] [UW] [W][AA] [W][AE]
[W][EH] [W][AO] [W][IY] [Y][AA] [Y][AE] [Y][EH] [Y][AO] [EU][IY]
[Y][OW] [Y][UW] # # }

#1
//    Transliterating all the Hangul Syllable Codepoints (AC00-D7A3)
[...]
CA03  {CAB} //
CA05  {CAD} //
CA01  {CAG} //
CA07  {CAH} //!
[...]

#2
//    Standardizes Hangul in a phonetically intuitive pseudolanguage
g}{X  _}{g
n}{X  _}{n
M}{X  r}{m
[...]
#3
//    The Basic Phonetic Rules in Korean are the Jamo
//    position rules expressed below
{<jamo><jamo><jamo>}    <initial-sound><medial-sound><final-sound>

#4
//    Sound to Sound rules
[AA][NG]    [AN] //     Vowel nasalization
[G][Y][EH]  [G][EH]
[...]
```

Appendix C: Korean PLI (sample)

```
//*********************************************************************
//*****************************English PLI*****************************
<vowel> = { a e i o u }
<vowel-sound> = { [AH] [EH] [IH] [AO] [UH] }
<consonant> = { b c d f g h j k l m n p q r s t v w x y z }
<consonant-sound> = { [B] [K] [D] [F] [G] [HH] [JH] [K] [L] [M] [N] [P]
[K] [R] [S] [T] [V] [W] [K][S] [Y] [Z] }

#1
//    Implements case insensitivity and transliterates
//    some borrowed characters.
0041  a   //A
0042  b   //B
0043  c   //C
[...]
0061  a   //a
0062  b   //b
0063  c   //c
[...]
00C9  e'  //É
00E9  e'  //é
00F1  ni  //ñ

#2
//    Stardardizes the many inconsistancies of English
+<consonant>ay+   +<consonant>[EY]+
+<consonant>ays+  +<consonant>[EY][Z]+
<vowel>ry+  <vowel>ree+
<consonant>ay+    <consonant>ey+
<consonant>ays+   <consonant>eyz+
+chair      +[CH][EH][R]
ause+ [AO][Z]+
+through+   +throo+
+how+ +h[AW]+
+off+ +[AO]f+
+four+      +faur+
+many+      +[M][EH][N][IY]+
+being      +bee[IH][N][G]
+into+      +intoo+
[...]

#3
//    Many English phoneme to sound rules present.
<vowel>     <vowel-sound>
<consonant> <consonant-sound>
aw    [AW]
edd   [EH][D]
[...]

#4
//    Deals with english Phoneme-to-Phoneme interaction
[EH][EY]    [IY]
[S][Z]      [S][EH][Z]
[L][F]+     [F]+
[...]
```

Appendix D: English PLI (sample)

```
//*************************************************************************
//**********************Serbo-Croatian PLI*********************************
//*************************************************************************

#1
//    Implements case insensitivity
//
0041  a  //A
0042  b  //B
0043  c  //C
[...]
//Native characters as found in Latin-B and padded start here
00E8  ch
[...]
00A9  sh

//Native characters as expressed in the Unicode Standard start here
[...]
01F3  dz
01C7  lj
01C8  lj
01C9  lj
01CA  nj
01CB  nj
01CC  nj
0160  sh
0161  sh
01E7  zh
01E6  zh
[...]
#2
//    Borrowed words here
wc    vetse

#3
//easy grapheme to sound mapping
a     [AA]
b     [B]
c     [TS]
d     [D]
[...]
aj    [AY]
ej    [EY]
oj    [OY]
y     [Y] //for borrowed words
#4
//non yet
```

Appendix E: Serbo-Croatian PLI (sample)

8. BIBLIOGRAPHY

[1] Carlson Ron, Bjorn Granstorm (1976), "Text-to-speech Based Entirely on Rules", ICASSP '76, 686-688.

[2] Clarkson, P. , Rosenfeld, R. (1997), "Statistical Language Modeling using the CMU-Cambridge Toolkit", EUROSPEECH '97, 2707-2710.

[3] Damper R.I. (1995), "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.

[4] Deng, L. (1997), "Integrated-multilingual Speech Recognition using Universal Features in a functional Speech Production Model", ICASSP '97, 1007-1010.

[5] Federking, R., Rudnicky, A., Hogan, C., (1997) "Interactive Speech Translation in the DIPLOMAT Project", Spoken Language Translation Worshop ACL- '97.

[6] Hertz, Susan (1982), "From text-to-speech with SRS", Journal of the Acoustical Society of America, 1155-1171.

[7] Lee, Kai Fu (1989), "Automatic Speech Recognition: The development of the SPHINX system", Klawer Academic Publishers.

[8] Lee Lin-Shan , Chiu-Yu Tseng, Hung-Yan Gu, F.H. Liu. C. H. Chang. S. H. Hsieh and C. H. Chen (1990), "A Real-time Mandarin Dictation Machine for Chinese Language with Unilimited texts and very large Vocabulary", ICASSP '90, 65-68.

[9] Matsuoka Tatsuo, Katsutoshi Ohtsuki, Takeshi Mori, Sadaoki Furui and Katsuhiko Shirai (1996) , "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.

[10] Placeway, P. et al (1997), "The 1996 Hub-4 Sphinx-3 System", Proc. DARPA Speech Recognition Workshop.

[11] Sejnowski, T.J. and Rosenberg, C.R. (1986), "Nettalk: a parallel network that learns to read aloud" The Johns Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01.

[12] Steeneken, H.J.M. and Lamel, L.F. (1994) "SQUALE : Speech Recognizer Quality Assessment for Linguistic Engineering", Proceedings ARPA Workshop on Spoken Language Technology, Plainsboro, New Jersey.

[13] Sproat, R., Multilingual Text Analysis for Text-to-speech synthesis

[14] Sangho Lee, Yung-Hwan Oh (1996), "A Text Analyzer for Korean text-to-speech Systems", ICSLP '96, Volume 3, SaP1L2.1.

[15] The CMU Dictionary:http://www.speech.cs.cmu.edu/comp.speech/Section1/Lexical/

[16] The Unicode Consortium (1996), "The Unicode Standard, version 2.0", Addison-Wesley Publishing Company.

[17] Van Coile, Bert M.J. (1989), "The DEPES Development System for Text-to-Speech Synthesis", ICASSP '89, 250-253.

[18] Weng, F.L, Bratt, H., Neumeyer, L., and Stolcke, A. (1997) A Study of Multilingual Speech Recognition", EUROSPEECH '97, 359-363.

Some of the aforementioned tools can be obtained at:

http://bertrand.speech..cs.cmu.edu/