A Generic Methodology Of Converting Transliterated Text To Phonetic Strings Case Study: Greeklish

Nikos Tsourakis[1] *and Vassilis Digalakis*[2]

[1] Dialogos Speech Communications S.A., Chania, Greece
[2] Department of Electronic and Computer Engineering, Technical University of Crete, Greece
ntsourak@speech.gr, vas@telecom.tuc.gr

Abstract

In this work, we present a generic methodology for converting transliterated text (native language written with a non-native alphabet) to phonetic sequences. The goal is to create the same phonetic result that would be produced if a native speaker uttered the original text in native alphabet. In our work, we implemented the specific methodology as a front-end to a Text-to-Speech (TTS) server. To evaluate our algorithms we considered the case that corresponds to the Greek language, called Greeklish.

Index Terms: automatic pronunciation generation, machine transliteration, classification and regression trees, text-to-speech

1. Introduction

Pronunciation modeling has been extensively studied for automatic speech recognition (ASR) and TTS systems – see for example the Johns Hopkins workshop on pronunciation modeling and the references therein [1]. In this paper we study pronunciation modeling for text written in a non-native alphabet.

Greeklish is a common way that people from Greece communicate using emails, instant messaging (like msn or icq), sms etc. In Greeklish, people transliterate Greek words using the Latin character set. The use of Greeklish originated from the lack of Greek character support in early software systems. It offers the freedom of writing text without the need of considering orthography and grammar rules or even punctuation. In the Greek language, where the punctuation of every single word is mandatory, one can save a lot of time by writing in Greeklish and omitting any punctuation, thus Greeklish is often used for writing e-mails or sending sms [2].

ELOT, The Greece's Standards Organization, has proposed a standard transliteration, which however has not been accepted by the general public [3]. For example, although the standard proposes that the Greek letter "θ" (theta) should be written with the diphthong "th", many people use the letter "u" instead (since on the Greek keyboards the letter "θ" appears in the position of "u"). Others, use the character "8" or "9", as each one of them resembles to the specific Greek letter [2][4]. Therefore, the development of an effective methodology for modeling the transformation from Greeklish to Greek is not trivial. We should note that the opposite transformation is straight forward, as one can follow the proposed standard.

Previous work tried to overcome this problem by using regular expressions techniques [5] or rule-based approaches [6]. The difference in our approach is that we propose an alternative representation of the input word as a sequence of phonemes. This representation will serve as an intermediate layer between the source and the target alphabet.

Our goal is to create a methodology that will not necessarily consider the orthography of the output text. We focus on a methodology that will return a phonetically acceptable output, since we want to create a front-end for handling Greeklish with TTS engines. Finally, since Greeklish text may contain pure English words, a mechanism for converting transliterated text to native alphabet should left the foreign words unaltered.

2. The Methodology

The methodology is a two-step approach. Each transliterated word is converted to a list of candidate pronunciations and each one of them is examined with the use of a set of dictionaries. The goal is to acquire the best possible result. The fundamental components of the methodology are the Pronunciations Manager and the Best-Result Engine.

2.1. Pronunciations Manager

The role of the specific module is to create a list of pronunciations for a given transliterated word. The pronunciations consist of strings of phonemes of the native language. The module uses a statistical pronunciation model of Greek words written in Greeklish. Given the model, the pronunciation generator seeks to create a list of the most probable outcomes. The model that we chose to integrate is based on classification and regression trees [7] and we shall refer to this module as Tri-Symbol CART (TSC).

TSC associates each letter of the given word, in the context that it appears, to the phonemes of the output language (in our case the Greek). The specific context is the letter that precedes and the one that follows the corresponding letter. The TSC module outputs probabilities for each letter within its context and for each associated phoneme.

The mapping from letters to phonemes could be applied to more than one consecutive letters or one letter can be mapped to more than one phonemes. This is essentially a way of associating multiple pronunciations (phonemes) to the same sequence of Greeklish letters.

The Greek phoneme set that we used consisted of 44 phonemes [8]. This set is specified using the Computer Phonetic Alphabet [9], which provides a system for easily expressing the phonemes in notation by the IPA (International Phonetic Alphabet - IPA) using a standard keyboard.

As an example, let us consider the Greeklish word "euro", uttered as "E v r o′". The extraction of its pronunciation is depicted in high level in Figure 1. The word is initially split to its different consecutive tri-symbols. The list of tri-symbols, associated with the given Greeklish word, is then fed into the model, which produces the most probable pronunciation.

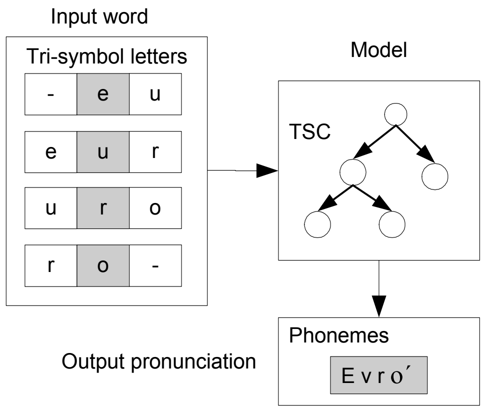

Figure 1: *Pronunciation generation*

The incorporated procedure is described briefly below:

- For a given word find the sequence of tri-symbol letters.
- With the use of the TSC model, calculate the probability of each sequence, utilizing every letter of the sequence in the context that it appears.
- Prune results below a predefined threshold.
- A list of pronunciations with their corresponding probability is now available in descending order.

2.2. TSC Model Design and Development

In the previous section we discussed the process of utilizing the model in order to obtain the pronunciations of a given word. We will now refer to the steps taken in order to create it. Although the procedure should be more or less the same for every application task, we will concentrate on our case study, the Greeklish. We will describe the design and training issues involved in the development of the TSC model.

2.2.1. TSC Model Design

The classification and regression trees let us categorize new data based on the data coded inside the structure of the tree. The specific structure is obtained after utilizing the training data and expresses the relationship between the attributes of the available data and classes. In our case, classification is based on the central letter and its left and right contexts.

During the expansion of the tree we begin with an initial node that contains all the training data and we proceed by making questions about the different attributes. As the tree expands, we want to obtain subsets that minimize the uncertainty. As an estimate of the uncertainty one can use the entropy (information gain criterion) defined in the equation below:

$$H(P(k)) = -\sum_{k} p_k * \log_2 p_k \qquad (1)$$

where k is the number of classes, in our case the Greek phonemes. Other estimates include the information gain ratio, the Gini index and the twoing criterion.

The pruning of sub-trees is the final step of the creation of our CART. A tree may have sections biased towards the training data and thus loose its generality and the ability to classify new data correctly. For this reason we used techniques such as the minimum-error pruning, the cost-complexity pruning and the pessimistic pruning in a series of comparison tests.

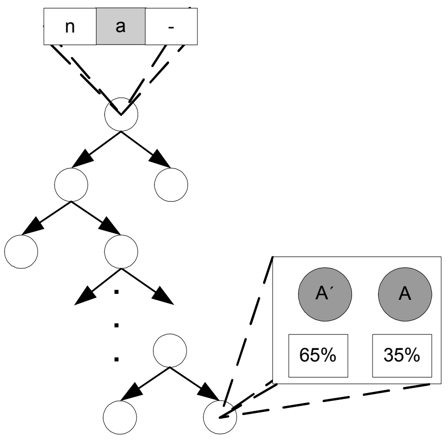

Figure 2: *Tri-Symbol CART*

An example of the usage of CART to create a statistical mapping of a tri-symbol to a phoneme is given in Figure 2. Each symbol is considered within its context and in respect to the different phonemes that it can be mapped to. Observe that the specific tri-symbol was mapped into two phonemes (classes) with different probability. The size of the corpus and the variety of the collected data is crucial in order to obtain unbiased statistics.

2.2.2. TSC Model Training

We initially gathered a significant number of texts containing Greeklish from multiple sources, such e-mails and the Internet. The unique letters encountered in the specific texts were 29. They correspond to the 26 letters of the English alphabet with the addition of “3”, “8” and “9”.

After removing any redundant punctuation marks, we manually mapped each Greeklish word with the corresponding Greek pronunciation. We thus obtained 8504 unique pairs of words and their associated pronunciations. The specific list was split into a training set, with 6000 pairs, and a test set of 2504 words.

The resultant dictionary (Greeklish words to Greek pronunciations) was used in order to train a Greeklish pronunciations model.

2.2.3. Letters-to-phonemes Alignment

In order to train the CART, we must associate each tri-symbol of the input word to the corresponding phoneme. This task can be time-consuming, especially when a large corpus is involved. We selected to solve the problem of aligning letters to phonemes automatically.

The idea is to create a trellis net, where the horizontal axis represents the phonemes and the vertical axis the letters (Figure 3). The number of letters corresponds to the letters of the input Greeklish word and the number of phonemes to the associated phonemes. For example, for m letters and n phonemes we would have an m x n net with nodes (i, j).

Starting from node (1, 1), the objective is to find the optimum path to node (m, n), which offers the best mapping to phonemes. From each node (i, j) there are three possible transitions:

- To node (i+1, j), which indicates that while moving to the next letter we remain in the same phoneme. Two or more consecutive letters can be mapped to the same phoneme.

- To node (i, j+1), which states that while moving to the next phoneme we remain in the same letter. One letter can be mapped to more than one phonemes.
- To node (i+1, j+1), which indicates that we are moving to the next letter and simultaneously to the next phoneme. These are the most frequent transitions.

As an example, consider the word “hill”, which is depicted in Figure 3. The black nodes represent the mapping to the corresponding phonemes. Observe that two consecutive letters (“ll”) are mapped to one phoneme.

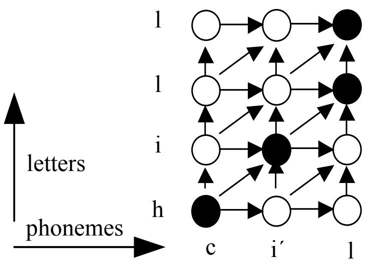

Figure 3: *Trellis net for the word “hill”*

We consider the best path as the most probable one and this is the criterion that will evaluate each path. The corresponding probability is the product of the probabilities of all the nodes that form a path ($P_i(j)$).

In order to calculate the probabilities $P_i(j)$, we perform a linear mapping from letters to phonemes using the Greeklish dictionary and count the frequency of mapping the phoneme j to letter i. The specific linear mapping is expressed by the following formula:

$$letter\_index = (\text{int})\frac{(phoneme\_index - 1)\cdot(number\_of\_letters - 1)}{number\_of\_phonemes - 1} + 1 \quad (2)$$

where *letter_index* is the letter to which the *phoneme_index* is associated to, *phoneme_index* is the phoneme that we want to associate*, number_of_letters* is the number of letters of the word and *number_of_phonemes* is the number of the Greek phonemes of the word.

After calculating the probabilities $P_i(j)$, we can now proceed to the search of the optimum path from node (1, 1) to node (m, n) using dynamic programming in order to obtain the best mapping of the word letters to their pronunciation phonemes.

2.3. Best-Result Engine

The best-result engine selects the most appropriate result within the list of pronunciations. It incorporates a set of dictionaries that are specific for the language that we examine and extracts the pronunciations contained in the specific dictionaries using hash methods. Four pronunciation dictionaries take part in the process:

- A Greeklish dictionary with 8504 words.
- A Greek dictionary with 93K stressed words. Notice that most words in Greek contain a stress mark.
- A Greek dictionary with the same 93K words in their unstressed version.
- An English dictionary with 120K words.

The execution steps of the engine are summarized below:

1. Initially a search in the Greeklish dictionary is performed and if the input word is found, the corresponding pronunciation is returned.
2. If a pronunciation is not found in the Greeklish dictionary, the pronunciation manager is engaged for the specific input word, producing a sorted list with the candidate pronunciations according to their probability.
3. Each one of the pronunciations is compared to the Greek pronunciation dictionary in descending probability order. The first one that is encountered, is returned as a valid pronunciation.
4. Step 3 is repeated using the unstressed Greek dictionary. The stressed version of the pronunciation is returned when a matching is achieved.
5. A search in the English dictionary is performed, hypothesizing that the input word is an English one.
6. Finally, if none of the previous steps succeeds, we return the first pronunciation from the sorted list, removing stresses in irrational places (e.g. 10 syllables before the end).

3. Pronunciation Model Evaluation Tasks

The output of the Greeklish pronunciation model is a list of pronunciations for a given word. During the evaluation task, we utilized the test set of 2504 unique Greeklish words and their corresponding pronunciations, which were fed to the pronunciations manager.

We chose to create five different models, which included:

- A minimum error tree using the twoing criterion and minimum-error pruning.
- A standard cart using the twoing criterion and performing cost-complexity pruning.
- A cart0 after creating a list of trees using cost-complexity pruning and selecting the tree with the minimum error. The standard tree is much simpler with an error comparable with the minimum.
- A standard C4 created using information gain ratio splitting and C4.5 pruning [10].
- A Pessimistic tree using twoing criterion and pessimistic pruning.

In order to evaluate the model we incorporated specific metrics, which included:

- The total number of correct pronunciations, without considering their place in the list.
- The average placement of the correct pronunciations in the list.
- The maximum number of the results contained in the list
- The number of leaves in the tree needed in order to achieve the values above.

According to the results demonstrated in Table 1, we can observe that for each of the 2504 words and with different model configurations, each tree produces a list with the candidate pronunciations in descending probability order. We determined that only 10 candidates should be produced, which is denoted in the first column (Result rank).

The entries in each row of the table are the number of the correct pronunciations found in that rank for the specific CART model.

For example, we can observe that for the standard cart model (third column), 2030 correct pronunciations were found, which gives us a percentage of 81.07%. The average placement of the results for the specific configuration is 1.42/10 while the maximum size of the results list is 10. The number of leaves in the CART for that case was 8338.

We can also see that we obtained the best result (81.87%) for the minimum error model and with 9478 leaves. We decided, however, to keep the model that corresponds to the standard cart model, which offers a similar performance with 1000 fewer leaves.

	Model				
Result rank	Min.	Stand.	Cart0	C4	Pess.
1	676	671	671	657	646
2	662	664	664	646	646
3	348	340	339	328	323
4	147	143	139	136	131
5	97	91	93	96	101
6	45	51	48	38	37
7	31	34	31	37	39
8	23	14	14	19	14
9	13	15	14	10	14
10	8	7	6	6	6
Total	2050	2030	2019	1973	1957
Percent. %	81.87	81.07	80.63	78.79	78.15
Average Placement	1.43 /10	1.42 /10	1.40 /10	1.41 /10	1.42 /10
Max. Res.	10	10	10	10	10
Leaves	9478	8338	8188	7635	7485

Table 1: *CART Model evaluation results*

4. Best-Result Engine Evaluation Tasks

During the evaluation of the best-result engine we considered two case studies. In the first configuration, the engine created up to 10 pronunciations for a given word, while in the second configuration we used only the top pronunciation. The test set was the dictionary utilized in the previous section with 2504 words. Using the procedure described in Section 2.3 (step 3-6), we came up with the word counts presented in Table 2 for each configuration and each step.

	Greek dictionary	Unstress Greek dict.	English dict.	First pronunc.
10 pron.	2018 (80.5%)	109 (4%)	10 (0.5%)	367 (15%)
1 pron.	713 (28.5%)	1263 (50%)	13 (0.5%)	515 (21%)

Table 2: *Best-result engine word-counts*

For 10 pronunciations we can deduce that 80.5 + 4% = 84.5% of the Greeklish words received a valid pronunciation. We should note that some of the 84.5% of the words that received a valid pronunciation (pronunciation of a Greek word) might have not received the correct one (the one we initially assigned on the Greeklish word). Similarly, some of the 15% of the words for which the algorithm chose the first pronunciation in step 6 may have indeed received the correct one. Our dictionaries are finite and therefore cannot be applied to all cases.

Similarly, in the case of the single pronunciation configuration the corresponding percentage of valid pronunciations is 78.5%.

A Greeklish word can be associated to multiple pronunciations, where the stress mark is the only thing that makes them distinct. In some cases, it is impossible even for a human to associate the correct pronunciation to a Greeklish word, without considering the context in which it appears. Therefore, the associated pronunciation of the test set is not necessarily unique and in some cases one should assign more than one.

We should finally mention that although the percentages are quite respectable, the end-user experience in a real application may be better than what the results indicate. A punctuation error will degrade the values of the metrics, but the end user will not necessarily notice the difference when the specific text is announced from a TTS (e.g. announcing e-mails written in Greeklish).

5. Conclusions

In this work we described a novel methodology for converting transliterated text to strings of phonemes. Although our discussion was concentrated on the Greek language, the methodology is generic and can be applied to most circumstances where an analogous problem arises. Hinglish (Hindi and English), Manglish (Malayalam and English) etc., are just other examples where our methodology can be applied.

The system is word-based and not text-based, which means that we cannot exploit the context in which a specific word is encountered. The system could be improved with the use of statistical models in order to process more elegantly the output. Moreover, a normalization step is necessary before the engagement of our method, in order to perform language detection, so that English text is not falsely considered as Greeklish.

6. References

[1] Johns Hopkins workshop on Pronunciation Modeling and Lexicon Adaptation for Spoken Language, 2002. "http://www.clsp.jhu.edu/pmla2002/cd/".

[2] T. Tseliga, "A corpus-based study of discourse features in Roman-alphabeted Greek (i.e. Greeklish) emails", International Conference on Internet and Language, Castellon, Spain, 2003.

[3] ELOT, Hellenic Organization for Standardization "http://www.elot.gr".

[4] D. Koutsogiannis and B. Mitsikopoulou, "Greeklish and Greekness: Trends and Discourses of 'Glocalness'" Journal of Computer-Mediated Communic. Vol I, 2003.

[5] A. Karakos, "Greeklish: An experimental Interface for Automatic Transliteration", Journal of the American Society for Information Science and Technology, 2003, pp. 1069-1074.

[6] "http://www.asda.gr/active/GrLish2.asp","http://www.translatum.gr/converter/greeklish-converter.htm".

[7] L. Breiman, J.H. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees, Wadsworth & Brooks, Pacific Grove, CA, 1984.

[8] V. Digalakis et al, "Large Vocabulary Continuous Speech Recognition in Greek: Corpus and an Automatic Dictation System", Proc. of the Interspeech, Geneva, Switzerland, 2003, pp. 1565-1568.

[9] The International Phonetic Association (IPA) "http://www.arts.gla.ac.uk/IPA/index.html".

[10] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1992.