NEW STRATEGIES FOR PRONUNCIATION BY ANALOGY

Tatyana Polyákova, Antonio Bonafonte

Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

ABSTRACT

The synthesis quality is influenced by many important factors, among which the correctness of the grapheme-to-phoneme (g2p) conversion is one of the crucial ones. Automatic letter-to-sound systems have been in the center of attention for the last decade. One of the most effective and promising methods resulted to be the so-called "pronunciation by analogy" method [8], based on the analogy in the grapheme context, allowing derivation of the correct pronunciation for a new word from the parts of similar words present in the dictionary. This paper aims at further development of this method. Novel scoring strategies for determining the best pronunciations were proposed. A word error rate reduction of 1.5-2.5 percent is obtained. A detailed analysis shows that one of the new strategies consistently outperforms the others. The results obtained are compared to other g2p methods using the same data.

Index Terms—Speech synthesis, grapheme-to-phoneme, pronunciation by analogy.

1. INTRODUCTION

The derivation of the pronunciation in English language given a letter string is a hard task for non-native speakers and it is even truer for automatic systems that are usually based on statistics. The human brain handles statistics in a different way; humans use analogy to memorize how to pronounce words or word fragments in English and other languages with deep orthography. When trying to read something, it takes time and extra effort to apply the pronunciation rules of the language, while the analogy matching that our brain performs is thunder fast. Either we say it or not correctly depend on the number of words with similar pronunciation rules that we have learned before. This is where the computer has a great advantage compared to, for example, English learners. For the computer, grasping all the examples from the dictionary and applying statistics-based analogy to derive pronunciation for the new words is a question of milliseconds. The pronunciation by analogy is an interesting technique similar to language learning that was successfully applied to derive pronunciation of out-of-vocabulary words [3, 8, 11]. One of the goals of this work was to compare the pronunciation-by-analogy system reported by Marchand and Damper [8] to other g2p methods reported in [9]. The possibilities of further improvement of the system's performance were explored by introducing new scoring strategies for choosing the best pronunciation.

2. PRONUNCIATION BY ANALOGY SYSTEM DESCRIPTION

For the first time, pronunciation-by-analogy (PbA) was proposed for reading studies by Glushko in 1979 [6] and later in 1986 Dedina and Nusbaum [3] introduced the use of this method to TTS applications. The latest and most successful implementation of the algorithm was published by Marchand and Damper [8] which we have reimplemented for our experiments. The system as well as the initial one, called PRONOUNCE [3] consists of four major components.

- Aligned lexicon (in one-to-one manner)
- Word matcher
- Pronunciation lattice (a graph that represents all possible pronunciations)
- Decision maker (chooses the best candidate among all present in the lattice)

Below we review the entire algorithm since it is necessary for understanding of the new strategies and introduction of new terminology.

In order to search for analogy between words that share similar substrings, in the first place it is necessary to make sure that there is a one-to-one match between the orthographic and phonetic strings, or, in other words, each letter has to be aligned to its corresponding phonetic representation. Finding the correct alignment is a challenge since the orthographic and phonetic representations of a word in English do not always have the same length. Due to its rather complex orthography, in English words there are usually more letters than sounds. In this case a null phone /_/ is inserted into the phoneme string, ex. #*thing*# /# T _ i N _ #/, otherwise, if the number of phonemes is greater than that of letters, the phonemes corresponding to the same letter are joint together in one, e.g. *fox* /f A k_s/. The alignment is based on EM algorithm, and it is similar to that described in [2]. The alignment given by the system is not always the correct one and it can influence negatively on the results.

After the dictionary has been aligned, the matcher, one of the most important components of the system, starts to search for common substrings between the input word and the rest of the dictionary entries. Every input word is then compared to all the words in the lexicon in order to find common "arcs". Let us call the substrings in the grapheme context "letter arcs" and the corresponding substring in the phoneme context "phoneme arcs". All the possible letter arcs with the minimum length of 2 letters and the maximum length equal to the input word length are generated and then searched in the dictionary. For every letter arc from the input word, matching with the same letter arc from a dictionary word, the corresponding pronunciation or the phoneme arc is extracted. The frequency of appearance of each phoneme arc corresponding to the same letter arc is stored along with the start position is for each arc. As an example, let us say that the word *top* is absent from our dictionary; the list of all possible letter arcs for this word can be given as *"#t, #to, #top, to, top, top#, op, op#, p#"*. Now let us suppose that in the lexicon we have the word *"#topping#"* with the pronunciation /# t A p _ I _ N #/, here the matcher finds the letter arcs *#t, #to, #top,* and *op*, with their corresponding phoneme arcs /# t/, /# t A/, /# t A p/, /A p/. Each

time that for the same letter arc we find the same phoneme arc; the frequency of the phoneme arc is incremented. The matching phoneme arcs are entered into the pronunciation lattice that can be represented by nodes and connecting arcs. If an arc starts at a position *i* and ends at a position *j*, and if there is yet no arc starting or ending at position *i*, the nodes L_i and L_j are added to the graph. An arc is drawn between them. All the nodes are labeled with the corresponding "juncture" phoneme and its position in the word. The arcs are labeled with the remaining phonemes and their frequency of appearance. An example of the lattice construction for the word top using the arcs found in the word *topping* is illustrated in Figure 1. All the arc frequencies are assumed to be equal to 1. Each complete path through the lattice is called "pronunciation candidate". We considered only the shortest paths through the lattice [8]. If there is a unique shortest path, it is chosen as the best pronunciation and the algorithm stops. Usually there are several shortest paths through the lattice, and a decision function is necessary to choose the best pronunciation candidate among them.

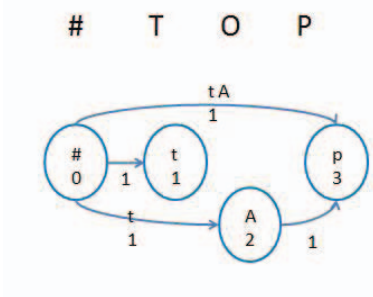

Fig. 1. *Lattice construction for the word" top".*

Each candidate can be represented as $C_j=\{F_j,Dj,P_j\}$, where $F_j = \{F_1,\ldots,F_n\}$ are the phoneme arc frequencies along the *jth* path, $D_j = \{d_1,\ldots,d_n\}$ are the arc lengths and $P_j = \{p_1,\ldots,p_l\}$ are the phonemes comprising the pronunciation candidate, being *l* the pronunciation length. Marchand and Damper in 2000 [8] proposed to use 5 scoring strategies in order to choose the best pronunciation. In the same work two ways of strategy combination were introduced. Each strategy gives us a score for each candidate and based on its score each candidate is assigned a rank. According to the rank, each candidate is awarded points. If a strategy gives the same score for several candidates, they are given the same rank and the same number of points. There are two manners of determining the winner candidate; the first one is the sum rule, which chooses the candidate that has the largest value of the sum of points for all of the included strategies. The product rule chooses the candidate with the largest value of product of the points awarded by each of the included strategies. For NETtalk dictionary the best accuracy obtained was equal to 65.5% for words and 92.4% for phonemes, using all five strategies [8]. The sum and the product rule seemed to give the similar results.

3. MULTI-STRATEGY APPROACH

The original 5 strategies [8] are:

1. Maximum arc frequency product (PF)
2. Minimum standard deviation of arc lengths (SDPS)
3. Highest same pronunciation frequency (FSP)
4. Minimum number of different symbols (NDS)
5. Weakest arc frequency (WL)

The proposed strategies are:

6. Weighted arc product frequency (WPF)
Similar to 1st strategy described in [8], where for each arc the corresponding arc frequencies are multiplied $PF(C_j) = \prod_{i=1}^{n} f_i$, *n* being the candidate length, or the number of arc of which the candidate consists. Rank 1 is given to the candidate scoring the maximum *PF()*. The difference is that in this strategy for each phoneme arc, A_k the frequency of its appearance is divided by *k*, the number of different phoneme arcs found in the dictionary for the corresponding letter arc, L_i. For example if our unknown word, is *#infinity#* and if in the pronunciation lattice we have a path that starts with a letter arc, $L_{i=}$ *"# in"* and a corresponding phoneme arc A_1=/# @ N/, whose frequency is equal to 12, in order to obtain the weighted arc frequency, we have to divide 12 by the number of different phoneme arcs available in the dictionary for the letter arc *"#in"*.

7. Strongest first arc (SF)
This strategy aims at capturing the analogy in prefixes. The candidate with the highest frequency score for the first arc is given rank 1.

8. Strongest last arc (SL)
This strategy is analogous to the previous one but for the suffixes. The candidate with the highest frequency score for the last arc is given rank 1.

9. Strongest longest arc (SLN)
The candidate who has at the same time the longest and the most frequent arc is given rank1. First the longest arc is chosen and if there is a tie the next step is to choose the most frequent one. The candidate that have the longest and arcs seem to be more reliable, and of course, the more frequent the arc is the stronger is the analogy.

10. Same symbols multiplied by arc frequency (SSPF)
The 10th strategy is similar to the fourth one (*NDS*). *NDS* gives preference to the candidates whose phonemes appear in the majority of other candidates. $NDS(C_j) = \sum_{i=1}^{l}\sum_{k=1}^{N}\delta(P_{j,i}, P_{k,i})$, being *l* is the number of phonemes in a pronunciation, δ the Kroneker delta, equal to 1 if $P_i \neq P_k$ and 0 otherwise, and *N* the number of candidates. In our strategy when counting the common phonemes, we also take into consideration the phoneme arc frequencies. For every candidate the pronunciation is. If a candidate has a common phoneme with other candidates, we give it a higher score, depending also on the number of times the phoneme arc containing that phoneme appears in the dictionary $SSPF(C_j) = \sum_{i=1}^{l}\sum_{k=1}^{N}\left(1-\delta(P_{j,i}, P_{k,i})\right) * f_{arc(i)}$.

11. *Product frequency, same pronunciation (PFSP)*
This strategy is a combination of 1st and 3rd strategies [8]. The 3rd strategy gives the privilege to the candidates sharing the same pronunciation with the others, rank 1 is given to the candidate scoring the maximum *FSP()*.
$FSP(C_j) = cand\{P_j | P_i = P_k\},\ j \neq k\ and\ k \in [1, N]$
In eleventh strategy all the candidates that share the same pronunciation obtain the same score equal to the combination of

the scores assigned to each one of the candidates by the 1st strategy $PFSP(C_j) = \sum_{\forall\, k, P_{k=P_j}} \sqrt[n]{PF(C_k)}$.

4. EXPERIMENTAL RESULTS

The experiments were performed on two dictionaries, NETtalk and LC-STAR, used by the authors in previous experiments [8, 9].
The NETtalk has 20K of words, and it was manually aligned by Sejnowski and Rosenberg [13]. LC-STAR is a public dictionary of U.S. English, created in the framework of LC-STAR project [7]; we have used only the common words (about 50 K). No homonyms were considered for the experiments. As usual, 90 percent of the lexica were used for training and 10 for test. The first thing to do was to find out how each strategy performed. The strategy mask is a binary string, where one means the strategy is included in the final result and 0 otherwise. The results for eleven strategies for both dictionaries are given in Table 1.

Strategy mask/ Dict	NETtalk		LC-STAR	
	Ph. acc.	W. acc.	Ph. acc.	W.acc.
10000000000	89.70%	57.48%	94.76%	73.59%
01000000000	88.00%	50.59%	92.68%	65.31%
00100000000	89.95%	59.06%	95.60%	79.34%
00010000000	90.27%	57.43%	95.53%	76.73%
00001000000	88.56%	53.75%	94.07%	71.44%
00000100000	89.69%	57.02%	94.96%	75.05%
00000010000	89.15%	55.84%	92.95%	66.17%
00000001000	87.92%	50.28%	94.46%	72.26%
00000000100	88.68%	54.01%	92.82%	65.23%
00000000010	89.99%	58.30%	94.95%	74.61%
00000000001	91.14%	**62.94%**	96.01%	**80.32%**

Table 1. *Word and phoneme accuracy for each strategy for NETtalk and LC-STAR dictionaries.*

From the results above we can see that the strategies give different performance for different dictionaries. The best strategy is the proposed eleventh strategy and the second best is the original 3rd strategy for both dictionaries. For NETtalk dictionary, two proposed and three original strategies made it to the top 5 strategy list while for LC-STAR dictionary the top 5 strategies included three proposed and two original ones.

In the next step we evaluated all possible strategy combinations. For our implementation of the 5 original strategies the best results were obtained for the combination of 1st and 3rd strategies "10100". The accuracy obtained for NETtalk lexicon was 63.04% words and 91.02% phonemes correct; and 80.94% words and 96.07% phonemes correct for LC-STAR lexicon. These results are slightly different from those reported in [8], as well as the scores obtained for each original strategy with our system, but we believe that it is due to the implementation nuances. The top 5 combination results including the proposed strategies are given in Tables 2 and 3.
Eleventh strategy is present throughout Tables 2 and 3 and its contribution to the improvement of overall score is the greatest for both lexicons. The best strategy combination results obtained are higher than those previously obtained combining only the original strategies. The word error rate decreased from 36.96% to 36.5% for NETtalk and for LC-STAR from 19.06% to 18.78%. That is between 1.5 and 2.5 percent of error decrease.

S. combination	Ph. acc.	W. acc.
11110010011	91.28%	**63.50%**
01110110011	91.24%	63.40%
01100010001	91.30%	63.40%
01100010011	91.29%	63.35%
00100010001	91.31%	63.35%

Table 2. *Top 5 strategy combination results for NETtalk dictionary.*

S. combination	Ph. acc.	W.acc.
00101000001	96.13%	**81.22%**
01100001001	96.08%	81.12%
01111100001	96.11%	81.04%
01101001001	96.04%	81.04%
00101001001	96.09%	81.04%

Table 3. *Top 5 strategy combination results for LC-STAR dictionary.*

The hypothesis is that the performance of the PbA algorithm is different for short and for long words. For this purpose the test dictionary was split into several dictionaries containing words of the same length. The lengths ranged from 3 to 17 letters per word. The words that had only two letters were added to the dictionary of 3-letter words. For the LC-STAR dictionary the distribution of words by length is a Gaussian with its mean situated approximately at the length 8. It is true for both training and test dictionaries.

As we expected, the performance of each strategy depends on the length of the word. This could be used to select a strategy. However, the new strategy 11 has happened to be the best in all the cases. When looking at word accuracy, in the great majority of the cases the eleventh strategy is the best. For word lengths equal to 6 and 7 letters the word accuracy is the highest. The strategies strongly disagree on very short and very long words. Word accuracy is higher for shorter words, since there are less phonemes and the probability of having at least one phoneme wrong is lower. For phoneme accuracy, the eleventh strategy gives the best results. The phoneme accuracy is high starting with 5 letter words and remains this way even for very long words, but like in the word accuracy case the strategies disagree for very short and very long words. The best phoneme accuracy results very obtained for the words consisting of 14 letters using the eleventh strategy.

Finally, Table 4 compares the results obtained with the PbA algorithm to the ones previously obtained in [9]. This comparison allows us concluding that PbA is one of the best g2p methods up to now.

Classifiers	baseline
DT	67.47%
FST	79.38%
HMM	47.54%
PbA	**81.22%**

Table 4. *Word accuracy for different g2p methods.*

The results above were obtained for the LC-STAR dictionary using decision trees (DT) [1], finite state transducers (FST) [5] and hidden Markov models (HMM) [10] and PbA [8] classifiers.

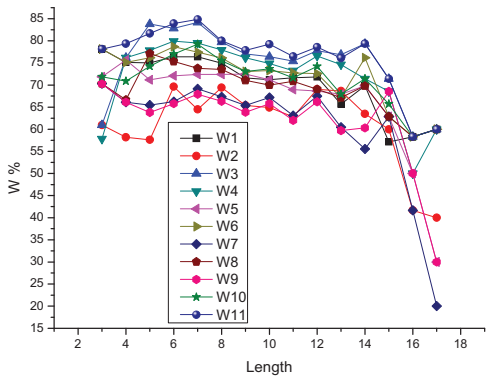

Fig.2. *Word accuracy for each strategy and word length.*

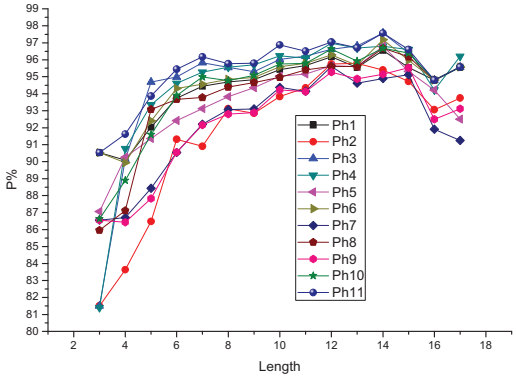

Fig.3. *Phoneme accuracy for each strategy and word length.*

5. CONCLUSIONS

For the pronunciation by analogy method new scoring strategies were proposed and improvements were obtained based on these strategies. The 1.5-2.5% of error reduction was reached in comparison with the strategies used in [8]. The proposed eleventh strategy was found to be the best one, as it can be seen from the results shown in Table 1 where it performs better than the other strategies on both dictionaries, as well as from Tables 2 and 3 where it participates in all top 5 strategy combinations. The difference between strategy combination results with or without the eleventh strategy is less significant than the difference between the accuracy obtained for the eleventh strategy and the rest of the strategies. The performance of each strategy on words of different length was analyzed; Figures 2 and 3 show that for all word lengths the eleventh strategy performs best both for words and phonemes.

6. ACKNOWLEDGEMENTS

This work was sponsored by the Spanish Ministry of Education (*AP2005*-4526) and AVIVAVOZ project.

7. REFERENCES

[1] Black A.W., Lenzo K. and Pagel V., "Issues in building general letter to sound rules", *In Proceedings of the Third ESCA workshop on speech synth*esis, Jenolah Caves, W-S W, Australia, pp. 77-80, 1998

[2] Damper R. I., Marchand Y., Marsterns J.-D. and Bazin A., "Aligning letters and phonemes for speech synthesis" *in Proceedings of the 5thISCA Speech Syntesis Workshop,* Pittsburgh, pp. 209-214, 2004

[3] Dedina, M. and Nusbaum, H. "Pronounce: a program for pronunciation by analogy", *Computer Speech and Language,* Prentice-Hall, London, UK., vol .5, pp 55—64, 1991

[4] ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/

[5] Galescu L., J. Allen, "Bi-directional Conversion Between Graphemes and Phonemes Using a Joint N-gram Model", *In Proc. of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Perthshire, Scotland, 2001

[6] Glushko, R. J., *"Principles for Pronouncing print: The psychology of phonography*. Lesgold and Perfetti", pp. 61—84, Lawrence Erlbaum, Hillsdale, NJ, 1981

[7] http://www.lcstar.org

[8] Marchand, Y. and Damper R.I. "A multi-strategy approach to improving pronunciation by analogy", *Computational Linguistics 26(2)*, pp. 195-219, 2000

[9] Polyakova T., Bonafonte A., "Learning from errors in grapheme-to-phoneme conversion", *International Conference on Spoken Language Processing*, pp.1149-1152, Pittsburgh, USA, 2006.

[10] Taylor P., "Hidden Markov Models for grapheme to phoneme conversion", *In Proc. of Interspeech 2005*, Lisbon, Portugal, pp. 1973-1976, 2005

[11] Yvon, F., "Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks", In *Proc. NeMLaP'96*, pp. 218-228, 1996