FURTHER IMPROVEMENTS TO 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 conversion is one of the crucial ones. The globalization phenomenon makes it impossible to have a dictionary with all of the existing words for each language. 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 the study of this method's performance and comparison to authors' previous work, furthermore novel scoring strategies for determining the best pronunciations were proposed along with new ways of their combination. A word error rate reduction of 1.5-2.5 percent was obtained.

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 in 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 apply 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 derived pronunciation of out-of-vocabulary words [4,8,11].

Another important aspect of language learning is learning from errors. When a new word is pronounced erroneously a new word and corrected by a native speaker or a teacher, our brain learns not to commit the same error in a similar situation. The more examples of similar errors, given a similar error occurrence situation we have, the better we learn not to commit the same error again.

Combining these two methods used by language learners powered up by the computer CPU's learning and computing capacity we are able to improve the grapheme-to-phoneme module. The objective of this work was to compare the pronunciation-by-analogy system reported by Marchand and Damper [8]. This paper presents an interesting contribution to the research in speech synthesis due to the comparison of the grapheme-to-phoneme methods using the same dictionaries for training and testing of the systems. The possibilities of further improvement of the system's performance were explored from different perspectives. New scoring strategies were proposed and new ways to combine strategies by applying error-driven learning were studied.

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 [4] 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 [4] 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)

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 [3]. 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 in the operational phase 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. Before the matching starts each word in the dictionary and its pronunciation are added word beginning and end marks, for example *#thing#* #T _ i N _ #/. 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 for 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, we can assume 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 the frequency of their 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 was unique shortest path, it was chosen as the best pronunciation and the algorithm stopped. In the usual case when there are several shortest paths through the lattice, it is necessary to choose the best pronunciation candidate among them. Therefore, the last but not least component of the algorithm is the decision making function.

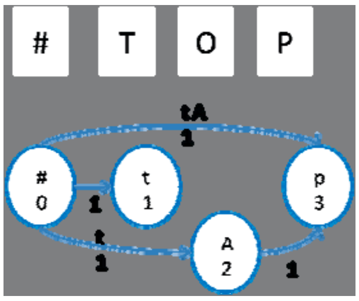

Figure 1. *Lattice construction for the word top.*

Each candidate can be represented as C_j={F_j,D_j,P_j} , where F_j = {F_1,…,F_n} are the phoneme arc frequencies along the jth path, D_j = {d_1,…,d_n} are the arc lengths and P_j = {p_1,…,p_l} are the phonemes comprising the pronunciation candidate, being l is the pronunciation length.

Marchand and Damper in 2000 [8] proposed to use 5 scoring strategies in order to choose the best pronunciation. They will be explained with in more detail in the next section. 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

In our work we have extended the study of the scoring strategies implemented 6 new scoring strategies. All of the scoring strategies, the original ones and the proposed ones involve phoneme arc frequencies f_i, arc lengths d_i, and p_l, the phonemes of which the candidate consists.

The original 5 strategies [8] are:

1. Maximum arc frequency product (PF)

For each arc the corresponding arc frequencies are multiplied $PF(C_j) = \prod_{i=1}^{n} f_i$, n is the candidate length, or the number of arc of which the candidate consists. Rank 1 is given to the candidate scoring the maximum PF().

2. Minimum standard deviation of arc lengths (SDPS)

$SDPS(C_j) = \sqrt{\frac{\sum_{i=1}^{n}(d_i - \bar{d})^2}{n}}$, where $\bar{d}$ is the median arc length. Rank 1 is given to the candidate scoring the minimum SDPS().

3. Highest same pronunciation frequency (FSP)

The privilege is given to the candidates that share the same pronunciation with the others $FSP(C_j) = cand\{P_j | P_i = P_k\},\ j \neq k \ and \ k \in [1, N]$, rank 1 is given to the candidate scoring the maximum FSP().

4. Minimum number of different symbols (NDS)

This strategy gives preference to the candidates whose phonemes appear in the majority of other candidates. $DS(C_j) = \sum_{i=1}^{l} \sum_{k=1}^{N} \delta(P_{j,i}, P_{k,i})$, where l is the number of phonemes in a pronunciation, δ is the Kroneker delta, which is equal to 1 if $P_i \neq P_k$ and 0 otherwise, and N is the number of candidates, rank 1 is given to the candidate scoring the minimum NDS().

5. Weakest arc frequency (WL)

The candidate whose lowest arc frequency value is the highest $WL(C_j) = \min_i\{f_i\}$, rank 1 is given to the candidate scoring the maximum WL().

The proposed strategies are:

6. Weighted arc product frequency (WPF)

Similar to Strategy 1, but 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 word, for which we are searching for the pronunciation is #infinity# and if in the pronunciation lattice we have a path that starts with a letter arc, $L_{1=}$ "# 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". Let us say that besides A_1 we also found the following phoneme arcs: A_2 = /# I N/ and A_3= /# _ n/. Then the weighted frequency for A_1 is $WF(A_1)$= 12/3

7. Strongest first arc (SF)

The seventh 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 tenth strategy is similar to the fourth one (NDS), but on one hand when counting the common phonemes, we also take into consideration the phoneme arc frequencies.
For every candidate the pronunciation is compared phoneme by phoneme to other candidate pronunciations.
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)*

The combination of first and third strategy, here all the candidates that share the same pronunciation obtain the same score, which is equal to the combination of the scores assigned to each one of the candidates by the first 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 dictionary, used by the authors in previous experiments.

The NETtalk has 20K of words, and it was manually aligned by Sejnowski and Rosenberg publicly available at [13]. The phonetic symbols used by Sejnowski and Rosenberg are left unchanged.

The 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). The phone set used is SAMPA. [10]. No homonyms were considered for the experiments. As usual, 90 percent of the lexicons 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%
00010000000	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. *Results for each strategy for NETtalk and LC-STAR dictionaries.*

From the results above we can see that the strategies give different performance of different dictionaries. The best strategy is the proposed strategy 11 and the second best strategy is the original strategy 3 for both dictionaries. For NETtalk dictionary 2 proposed strategies and 3 original ones made it to the top five strategy list while for LC-STAR dictionary the top five strategies include 3 proposed and 2 original ones. As the next step we evaluated all possible strategy combinations, in the strategy combination mask 1 means the strategy is included in the final decision and 0 the strategy is left out.
For our implementation of the 5 original strategies the best results obtained for NETtalk lexicon were 63.04% words and 91.02% phonemes correct, given the combination of first and third strategies "10100" and 80.94% words and 96.07% phonemes correct for LC-STAR lexicon and the same strategy combination. 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 best word accuracy obtained in [8] is 65.5% using all five strategies for NETtalk lexicon.The top five combination results are given in Tables 2 and 3.

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 five strategy combination results for NETtalk dictionary.*

As before, the top five results include proposed strategies. Eleventh strategy is present throughout Tables 2 and 3 and its contribution to 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's between 1.5 and 2.5 percent of error decrease.

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 five strategy combination results for LC-STAR dictionary.*

To further explore the possibilities of improvement for grapheme-to.-phoneme scores the transformation based learning was applied to strategy combination.

The transformation-based error-driven algorithm (TBL) originally invented by Eric Brill [2] consists in learning the transformation rules from the training data that is labeled with some initial classes. Using the TBL algorithm to correct the prediction previously obtained by another classifier allows us to capture the imperfections of previous approximations to the linguistic irregularities into a set of context-dependent transformation rules, where the context serves as the conditioning features.

In our case we took the best combination of original strategies as the initial prediction for LC-STAR dictionary to correct. The additional features were letters, phonemes and also each original strategy prediction per experiment. In order to obtain training predictions, we used n-fold evaluation, with n equal to the number of words in the dictionary. Each nth word was removed from the dictionary and input as the unknown word to the "pronunciation by analogy" system.

The best additional feature was found to be the "00100" or the third prediction standing alone, and it gave 81.46% words and 96.21% phonemes correct, using the 4-letter context and no constraints for correction. Constraints would limit the algorithm to correct the erroneous phonemes only by the ones previously seen in the training data. Using fourth and fifth predictions gave a slighter improvement up to 81.04% and 81.12% words correct correspondingly. These results should be compared to 80.94%, best result using only the original strategies. We have also used all five predictions as additional features but the improvements were not significant.

The results show that pronunciation by analogy captures very well all the regularities in English orthography, not leaving much room for improvement for the TBL method.

Comparing these results to previously obtained in [9], shown in Table 4 we can conclude that PbA is the best grapheme-to-phoneme method up to now.

Classifiers	baseline
DT	67.47%
FST	79.38%
HMM	47.54%
PbA	**80.94%**

Table 4. *Word accuracy for different grapheme-to-phoneme 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 classifiers.

5. CONCLUSIONS

This paper gives an overview of the pronunciation by analogy method used for g2p. New scoring strategies were proposed and the 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 transformation-base learning algorithm was applied and the results were analyzed. New strategy combination methods were considered and slight improvements attained. The fact that applying rule-based error correction did not give important improvements allows concluding that the PbA methods is capable of capturing quite well the regularities in English orthography.

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 synthesis, Jenolah Caves, W-S W, Australia, 1998

[2] Brill E., "Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging", Computational linguistics 21(4), pp. 543-565, 1995

[3] 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, 209-214., 2004

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

[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

[8] Glushko, R. J., "Principles for Pronouncing print: The psychology of phonography. Lesgold and Perfetti", 1981

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

[8] Marchand, Y. and Damper R.I. "A multi-strategy approach to improving pronunciation by analogy",Computational Linguistics26(2)pp. 195-219, 2000

[9] Polyakova T., Bonafonte A., "Learning from errors in grapheme-to-phoneme conversion", International Conference on Spoken Language Processing, 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

[12] http://www.phon.ucl.ac.uk/home/sampa/

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