English Phonetic Synthesis Based on DFGA G2P Conversion Algorithm

HongLin Chen*
Fuzhou University of Foreign Studies and Trade, China, 350007
*Corresponding author e-mail: 2561162169@qq.com

Abstract. In English phonetic synthesis, it is impossible to create a thesaurus containing all vocabulary as English has an almost unlimited vocabulary. Hence, for English words that are not included in the thesaurus, generating phonetic symbols through the “Grapheme-to-phoneme (G2P)” algorithm is the best solution. For this purpose, a dynamic finite generalization (DFGA) machine learning algorithm for the rules of G2P conversion is proposed in this paper. The dictionary library used for learning has 27,040 words, 90% of which are used for rule learning, and the remaining 10% are used for testing. After ten rounds of cross-validation, the average grapheme conversion accuracy in the learning and test sets is 99.78% and 93.14%, and the average vocabulary conversion accuracy is 99.56% and 73.51%, respectively.

Keywords: Phonetic Synthesis, Grapheme-to-phoneme Conversion, Machine Learning, Limited Generalization

1. Introduction
In English phonetic synthesis, “Grapheme-to-phoneme” is an essential link. In phonetic synthesis, to synthesize the phonetic of a word, it is required that the pronunciation should be known first. For English words, you need to know its phonetic symbol. Because English has almost unlimited vocabulary, no matter how large the vocabulary is, there may always be some words that cannot be found in the vocabulary during phonetic synthesis. How to solve the pronunciation of these words becomes an English phonetic synthesis system issue to be solved [1]. However, the problem is that the pronunciation of English words is very complicated. How a word is pronounced may be related to multiple factors, such as part of phonetic (such as the pronunciation of records as nouns and verbs), morphological structures (such as academic and academic, because (The vowels a and e are pronounced differently at the end of the word.) And Etymology [2]. However, the most important determinant is the collocation relationship between the letters that make up the word. The reason is that in most cases, if the spelling of two words is similar, the pronunciation is also
similar. On this basis, this paper will try to study the pronunciation rules by the collocation relationship between the letters in a word.

In the final analysis, the conversion of words into phonemes is to convert letters into corresponding phonemes (Letter-to-Phoneme), but the problem is that in modern English, there is no one-to-one correspondence between letters and phonemes, and a letter can be more than one corresponding to phoneme, such as x can be /ks/or/gz/; at the same time, a phoneme can also correspond to multiple letters, such as the score of or in the score /ɔ/. This brings trouble to the conversion of words into phonetic symbols because if there is no one-on-one correspondence between grapheme and phoneme, it is challenging to convert effectively [3]. Therefore, we must first establish one-on-one correspondence between grapheme and phoneme. This correspondence is more representative of two types: (1) the direct correspondence between grapheme and phoneme, that is, directly established between grapheme and phoneme. In the correspondence for grapheme or phoneme without correspondence, it is represented by the substitute “-”, (2) The correspondence between grapheme and phoneme, this correspondence is based on the actual pronunciation rules of English to establish one-on-one correspondence between grapheme and phoneme. For example, the one-on-one correspondence between grapheme and the phoneme in the word afterthought [4].

In the research work, the second correspondence relationship will be used, namely the one-to-one correspondence between the phoneme and the phoneme. Hence, “G2P symbols” can be split into three sub-tasks, namely (1) grapheme segmentation, that is, studying how to correctly segment an English word into graphemes, such as dividing the word afterthought into aft- er-th-ou-gh-t; (2) G2P conversion (G2P); (3) accent annotation, that is, studying how to correct accents and sub-accents [5-6].

It is assumed in this paper that all graphemes have been defined, a total of 127 graphemes, and the correspondence between graphemes and phonemes has been defined. For example, the grapheme u can correspond to 12 phonemes. Meanwhile, the problem of grapheme segmentation has been successfully solved, that is, any English word can be correctly segmented into graphemes. The problem to be solved is the second sub-task mentioned above, namely, how to convert the graphemes into phonemes correctly. In this paper, the case of polysyllabic words will not be considered (the pronunciation of polysyllabic words will be left to the polysyllabic disambiguation module), and no other information (such as part of phonetic, number of words, etc.) is collected. Study the collocation relationship between letters in a word to study the regularity of the G2P conversion.

The dictionary library for learning has 27040 words, which basically covers some of the most commonly used words in English. To learn and test the G2P conversion rules, based on the “Concise English-Chinese Concise Dictionary” in advance, all the words are grapheme-separated by hand, and after careful proofreading and checking, to ensure that all the scoring is complete Probably correct.

2. Description of learning task
Obviously, the ultimate goal of learning the G2P conversion rules is that for any English word (such as ea-sy) that has been split into graphemes, it can correctly convert the grapheme into the corresponding phoneme (such as (i : -z)]), where the “-” symbol is used to separate the phoneme and phoneme). Therefore, the entire learning task can be split into a series of subtasks, that is, learning the conversion rules for each grapheme, and if a grapheme corresponds to more than one phoneme, such as ea corresponds to/i:/,/e/,/ei/Etc., can be divided into smaller subtasks, that is, the rule learning of a certain
grapheme into a specific phoneme, such as the learning rule of the grapheme ea into /i:/.

The rule learning where the grapheme ea is converted into /i:/ is taken as an example to describe the task. To know under what circumstances ea is converted into /i:/, it is actually to find such a target function (Target Function), as shown in equation (1):

$$c(X) = \begin{cases} 1, & \text{Represents ea converted to } /i:/ \\ 0, & \text{Indicates ea no conversion } /i:/ \end{cases}$$ \hspace{1cm} (1)

Where \(x\) is a vector composed of some eigenvalues of the grapheme ea, defined as follows:

$$x = (x_1, x_2, \ldots, x_n)$$ \hspace{1cm} (2)

The word appeasement is taken as an example, which is segmented into the graphemes a-pp-ea-se-ment. If two graphemes are taken before and after the morphem ea as the feature values, the corresponding feature vector \(x = (a, pp, s, e)\).

Hence, the task of the rule learning that the grapheme ea is transformed into the phoneme /i:/ is to find such a function \(c\). However, in practical applications, it is difficult to accurately obtain such a function, and often only an approximate objective function can be obtained through a learning algorithm, that is, a function approximation of the objective function is obtained. To obtain this approximate objective function, we extract the feature values of all words containing the grapheme ea from the thesaurus to form a training instance set. Based on this training instance set, the approximate objective function is solved by machine learning algorithms. Table 2 lists 7 examples where ea reads /i:/ in 5 instances and is identified by Pi (i is used for serial number), and ea in the other 2 does not read /i:/ . Read/e/ /ei/, etc., with Ni.

The concept suggests whether ea reads /i:/, which is 1, then reads /i:/; if it is 0, then does not read /i:/. Each training instance is an ordered pair \(<x, c(x)>\), where \(x\) is defined by equation (2) and \(c(x)\) is defined by equation (1).

3. Dynamic finite generalization (DFGA)

Finite Generalization Algorithm (Finite Generalization Algorithm) is an inductive concept learning algorithm. The algorithm is used to learn the grapheme segmentation successfully, where the segmentation accuracy in test cases can reach 97.88%. It is assumed that FGA is used for 7 examples.

However, the conversion of grapheme/phoneme is more complicated than graphemes segmentation. It is impossible to generate examples by extracting a fixed number of fixed-position features. Therefore, some changes are made to the FGA algorithm, and the improved algorithm is known as DFGA. The name of dynamic FGA comes from the fact that compared with FGA, DFGA is dynamic in the following three aspects:

(1) Learning objects are dynamic
(2) The number of eigenvalues is dynamic
(3) The position of the eigenvalue is dynamic

For the prefix, the extracted feature value can of course only be the grapheme behind it; the suffix feature value can only be the preceding grapheme, assuming the number of feature values is 4, then dis
and full are extracted from the word distrustful. The eigenvectors are all (tr, u, s, t).

For general graphemes, to give full play to the role of each feature value, it will be dynamically extracted according to the location of the learning object. The grapheme ea converted to /i:/ is taken as an example. It is assumed that the number of eigenvalues is 3, four cases may occur, as shown in Figure 2 (a), which are as follows: (1) ea is at the end of the word, so take the feature value as the first 3 graphemes, such as guinea, and the corresponding instance is <(gu, i, n), 1>; (2) ea is in the penultimate position, then take the first 2 graphemes and the next 1 grapheme as the feature value, such as blockhead, the corresponding instance is <(ck, h, d), 0>; (3) ea is at the beginning of the word, then take the next 3 graphemes as the feature value, Such as eastward, the corresponding instance is <(s, t, w), 1>; (4) The other cases fall in the same category, such as the instance corresponding to the grapheme ea in beast is <(b, s, t), 1>; The corresponding instance of ea in alert is <(r, d, y), 0>. And if the number of eigenvalues is set to 4, there are 5 possible cases.

When extracting feature values, the right grapheme is always prioritized, because research shows that the right grapheme is more important for determining the pronunciation of the current grapheme.

4. Grapheme/phoneme conversion learning based on DFGA

After the DFGA is introduced, how to learn G2P conversion rules based on DFGA is explored. Among the 127 graphemes, the pronunciation of 53 graphemes is unique. For example, the pronunciation of the graphemes bb is always/b/, and the pronunciation of ck is always/k/. The above pronunciation, obviously, there are only 74 of these graphemes that need to be converted. Among the 74 graphemes, not every grapheme and its corresponding phoneme need to be learned. According to the statistical results of the thesaurus, a pronunciation with the highest frequency is used as the default pronunciation. For example, the grapheme gg. There are two possible pronunciations, one is to read/g/, which is presented 122 times in the thesaurus, such as e-gg, ma-gg-ot, etc.; the other is to read/dʒ/, which only appears 12 in the thesaurus. Times, such as exa-gg-erate, su-gg-est, etc. Hence, for the grapheme gg, only the rules that are converted to/dʒ/ are learned, and when the specific application rules are converted, if the instance of the grapheme gg extracted from a word matches some rules, the grapheme gg is explained and pronounced as /dʒ/; otherwise, if no rule is matched, it is considered to be/dʒ/.

In the specific test, K-Fold Cross Validation is used, so that K is equal to 10, that is, 27 040 words are divided into 10 equal parts, each with 2,704 words. A total of 10 rounds of validation are performed. In each round of validation, 9 sets of words are used as the training set, and the remaining 2,704 are used as the test set, that is, 90% of the words are used for each round of learning and the remaining 10% are used for the test. The results from the 10 rounds of validation are shown in Table 1.

| Table 1. Results of 10 rounds of cross-validation |
|-----------------------------------------------|
| Learn                                      | test   |
| Average number of words        | 24336  | 2704   |
| Average word conversion accuracy | 99.56% | 73.51% |
| Average number of pixels        | 145382 | 16356  |
| Average grapheme conversion accuracy | 99.78% | 93.14% |
| Average number of rules         |        | 4205   |
The same instance library and IVSA (The treated Version Space Algorithm, an iterative variant space method, which is a machine learning algorithm and successfully applied to the learning of the phoneme/phoneme conversion rules) are used. Table 2 shows the comparison between the conversion rule learning and the test results using DFGA.

**Table 2. Comparison of morphological learning test results based on DFGA and IVSA**

| Study method | Average word conversion accuracy | Average grapheme conversion accuracy | Average number of rules |
|--------------|---------------------------------|-------------------------------------|-------------------------|
| DFGA         | 73.51%                          | 93.14%                              | 4205                    |
| IVSA         | 69.89%                          | 91.29%                              | 6736                    |

Table 2 shows that in the learning of G2P conversion rules based on DFGA, the conversion accuracy is higher, and the average number of rules is much less than that based on IVSA.

5. Conclusions

The English pronunciation system is highly complicated. For this reason, G2P has always been a challenging problem in English phonetic synthesis systems. In this paper, a dynamic finite generalization method (DFGA) is proposed, where the algorithm is used to learn the G2P conversion rules. After ten rounds of instance learning and testing, the average G2P conversion accuracy in the learning set and the testing set is 99.78% and 93.14%, and the average vocabulary conversion accuracy is 99.56% and 73.51%, respectively. Hence, it has achieved relatively good results.

**References**

[1] Nose, Takashi, Arao, Yusuke, Kobayashi, Takao. Sentence Selection Based on Extended Entropy Using Phonetic and Prosodic Contexts for Statistical Parametric Speech Synthesis[J]. IEEE/ACM Transactions on Audio Speech & Language Processing, 2018,25(5):1107-1116.

[2] Holzapfel Martin. HMM-based database segmentation and unit selection for concatenative speech synthesis[J]. Journal of the Acoustical Society of America, 2017, 105(2):379-382.

[3] Nose, Takashi, Chunwijitra, Vataya, Kobayashi, Takao. A Parameter Generation Algorithm Using Local Variance for HMM-Based Speech Synthesis[J]. IEEE Journal of Selected Topics in Signal Processing, 2017,8(2):221-228.

[4] Huynh D Q. Homograph filter for speech synthesis system[J]. Journal of the Acoustical Society of America, 2015, 109(4):1285-1296.

[5] Mohsen Maraoui, Naim Terbeh, Mounir Zrigui. Arabic discourse analysis based on acoustic, prosodic and phonetic modeling: elocution evaluation, speech classification and pathological speech correction[J]. International journal of speech technology,2016, 21(4):1071-1090.

[6] Levent M. Arslan, David Talkin. Codebook based face point trajectory synthesis algorithm using speech input[J]. Speech Communication, 2017, 27(2):81-93.