APSIPA Transactions on Signal and Information Processing, 2023, 12, e15

Original Paper

BASPRO: A Balanced Script Producer for Speech Corpus Collection Based on the Genetic Algorithm

Yu-Wen Chen[1,2], Hsin-Min Wang[3] and Yu Tsao[1,4*]

[1] *The Research Center for Information Technology Innovation, Academia Sinica, Taiwan*

[2] *The Department of Computer Science, Columbia University, United States*

[3] *The Institute of Information Science, Academia Sinica, Taiwan*

[4] *Jointly Appointed Professor with the Department of Electrical Engineering, Chung Yuan Christian University, Taiwan*

ABSTRACT

The performance of speech-processing models is heavily influenced by the speech corpus that is used for training and evaluation. In this study, we propose *BAlanced Script PROducer* (BASPRO) system, which can automatically construct a phonetically balanced and rich set of Chinese sentences for collecting Mandarin Chinese speech data. First, we used pretrained natural language processing systems to extract ten-character candidate sentences from a large corpus of Chinese news texts. Then, we applied a genetic algorithm-based method to select 20 phonetically balanced sentence sets, each containing 20 sentences, from the candidate sentences. Using BASPRO, we obtained a recording script called *TMNews*, which contains 400 ten-character sentences. *TMNews* covers 84% of the syllables used in the real world. Moreover, the syllable distribution has 0.96 cosine similarity to the real-world syllable distribution. We converted the script into a speech corpus using two text-to-speech systems. Using the designed speech corpus, we tested the performances of speech enhancement (SE) and automatic speech recognition (ASR), which are

*Corresponding author: Yu Tsao, yu.tsao@citi.sinica.edu.tw.

Received 13 August 2022; Revised 12 December 2022
ISSN 2048-7703; DOI 10.1561/116.00000155

one of the most important regression- and classification-based speech processing tasks, respectively. The experimental results show that the SE and ASR models trained on the designed speech corpus outperform their counterparts trained on a randomly composed speech corpus.

Keywords: Corpus design, Mandarin Chinese speech corpus, phonetically balanced and rich corpus, recording script design, genetic algorithm.

1 Introduction

Speech corpus plays a crucial role in the performance of speech-processing models. The speech corpus that is used to train and evaluate these models significantly affects their performance in real-world environments. Recently, massive amounts of data have been generated and collected. Therefore, models are often trained using a large amount of data to achieve better performance. However, not all research institutions can support such computing resources. Furthermore, the use of large amounts of data in listening tests to evaluate models is expensive and time consuming. Moreover, for personalizing models, the amount of data that can be collected from new users is often limited. Therefore, a representative speech corpus is essential for training and testing.

Active learning [4, 16] is a popular strategy used for training data sampling and selection. Active learning algorithm dynamically selects a subset of samples with labels that are most beneficial to improving the model during training. In this study, however, we focus on an algorithm that finds a fixed representative training and testing speech corpus for general speech-processing models. That is, active learning selects a corpus for a specific model to optimize it, whereas the proposed algorithm creates a model-independent corpus. The proposed algorithm can cooperate with active learning. Specifically, the model can be initially trained using the proposed representative corpus, followed by active learning to select the most beneficial samples for further training.

A representative speech corpus is often referred to as a phonetically balanced or rich corpus. Phonetic balance means that the frequencies of phonemes in the corpus are distributed as close as possible to the frequencies in real-world conditions, and a phonetically rich corpus implies that the dataset should cover as many allowed phonemes as possible. In previous studies, researchers have developed corpora of this type for multiple languages, such as Amharic [1], Arabic [2], Bangla [3], Urdu [22], Thai [33], Turkish [5], Mexican Spanish [28], Romanian [24] and Chinese [15, 30, 35].

Previously, phonetically balanced and rich corpora were designed by experts with linguistic backgrounds [12, 13, 37]. The experts manually wrote or chose sentences that could form a phonetically balanced corpus. However, creating

a phonetically balanced and rich corpus in this manner is time-consuming and difficult. In addition, sentences written by the same person tend to be similar and lack variation. Moreover, this method cannot be used to generate corpora for specific knowledge domains.

Automatic methods have also been proposed, in addition to manual development. Automatic methods usually begin with a large collection of sentences. An algorithm then selects sentences from the collection to form a corpus that meets these requirements. Selecting the desired set of sentences is an NP-hard set-covering optimization problem. In other words, evaluating all possible sets of sentences is computationally too complex to be solved within an acceptable time. To automatically compose a phonetically balanced corpus, [19, 24] proposed random sampling and evaluating sentence groups and chose the one that best meets the requirements. [1] and [30] proposed two-stage methods. The first stage selects important sentences that contain as many syllables as possible or consist of units that appear less frequently in the corpus. The second stage involves selecting sentences that can achieve the desired statistical distribution. Additionally, [29] used the perplexity of each sentence as an indicator to generate a corpus. Most automatic methods are based on greedy algorithms [3, 5, 15, 22, 35]. Genetic algorithms (GA), a well-known approach for solving NP-hard problems, on the other hand, have not received much attention in speech corpus development.

In [27], the authors proposed a GA-based method to automatically form a phonetically balanced Chinese word list; nevertheless, this study focused on word lists rather than sentence lists. Only a few previous studies have used GA to automatically select sentence sets [17, 18]. Moreover, these GA-based methods focus on phonetic and prosodic enrichment rather than phonetic balance and enrichment. The development of GA-based Chinese speech corpora has not yet been thoroughly investigated.

Mandarin Chinese is a tonal syllabic language with five different tones, including four main tones and a neutral tone. Syllables that do not consider tone are denoted as base syllables. On the other hand, syllables that consider the tonal information are referred to as tonal syllables. Each syllable comprises an INITIAL (consonant) and a FINAL (vowel) and is represented by the pinyin system. The INITIAL and FINAL can be further decomposed into smaller acoustic units such as phonemes. Compared to phonemes, syllables are more intuitive to Mandarin Chinese speakers and are used more frequently. Therefore, we developed a tonal syllable-balanced and -rich (hereafter referred to as syllable-balanced) corpus to represent a phonetically balanced and rich corpus.

In this study, we propose an automatic method called BAlanced Script PROducer (BASPRO)[1] to compose a syllable-balanced Mandarin Chinese speech corpus. First, BASPRO uses pretrained natural language processing

[1]The toolkit is available via: https://github.com/yuwchen/BASPRO

(NLP) systems to extract candidate sentences from a huge Chinese news text corpus. Subsequently, a syllable-balanced recording script is generated using a GA-based method. Finally, the script is converted into a speech corpus using two text-to-speech (TTS) systems. The syllable-balanced recording script developed in this study is called *TMNews*[2] because the sentences in the script are collected from Mandarin Chinese news articles collected in Taiwan.

The contributions of this study are as follows.

- We propose BASPRO, which uses machine-learning-based NLP tools to process and extract candidate sentences from a collection of news articles.
- BASPRO employs a GA-based method to form a syllable-balanced recording script from candidate sentences. Experimental results show that the proposed BASPRO system can effectively select sentences according to the designed optimization criteria.
- The proposed BASPRO system is flexible in terms of language, data domain, and script size. In addition, it allows the generated script to have multiple sets, each satisfying the desired requirements. For example, in this work, each of the 20 sets is syllable-balanced, and the sentences do not overlap between sets.
- We analyze the performance of speech processing models trained on syllable-balanced (produced by BASPRO) and randomly composed speech corpora. Experimental results show that the speech processing models trained on the syllable-balanced corpus perform better than those trained on the randomly composed corpus.

2 The Proposed BASPRO System

The proposed BASPRO system consists of three main phases: data processing, script-composing, and postprocessing. The input is articles crawled from the Internet, and the output is a syllable-balanced recording script. Speech corpora can be generated from recording scripts using TTS systems or by asking people to make recordings. Figure 1 shows a schematic of the BASPRO system. First, the data processing phase extracts candidate sentences from the collected news articles. Simultaneously, the syllable distribution of the collected articles was calculated, which is denoted as real-world syllable distribution. The script-composing phase then generates a temporary syllable-balanced script from the candidate sentences. Finally, the postprocessing phase replaces unwanted sentences in the temporary script and produces the final script.

[2]The script is available via: https://github.com/yuwchen/BASPRO/tree/main/TMNews

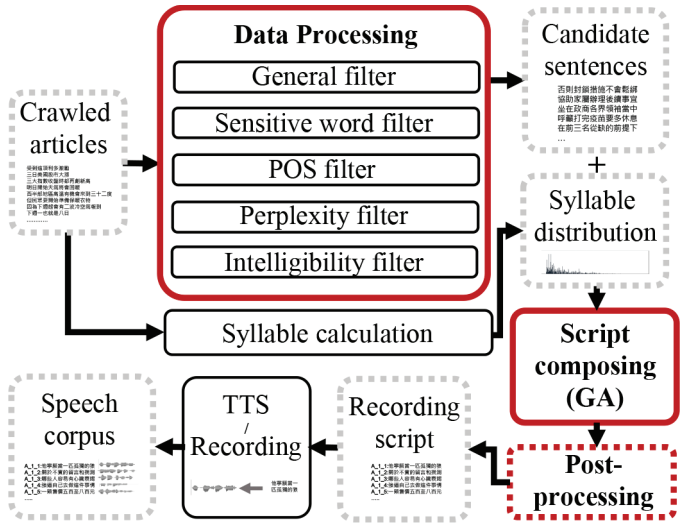

Figure 1: Schematic diagram of the proposed BASPRO system. In the data processing phase, candidate sentences were extracted from the collected articles. The script-composing phase uses real-world syllable distribution to compose a syllable-balanced script from the candidate sentences. Finally, the postprocessing phase replaces unwanted sentences in the script and produces the final script.

2.1 Data Processing

In the data processing phase, the input is news articles crawled from the Internet, and the output is candidate sentences. All sentences in the recording script were selected from the candidate sentences. We used five filters in the data processing phase to extract candidate sentences: (1) general, (2) sensitive word, (3) part-of-speech (POS), (4) perplexity, and (5) intelligibility filters. The general filter removes sentences with non-Chinese characters and keeps sentences with exactly ten characters. The sensitive filter then removes the sentences containing sensitive words. In this study, we let the sentences have a fixed length and excluded sentences containing sensitive words, as these settings are often required for listening tests. In addition, we designed a POS filter, perplexity filter, and intelligibility filter to filter out incomprehensible sentences. Because the resulting corpus will be used for listening tasks, we do not want any sentences to be difficult to understand and thus affect the evaluation results.

The POS is a category of lexical items with similar grammatical properties. Words assigned to the same POS often play similar roles in the grammatical structure of a sentence. We used POS as an indicator to exclude sentences that may not be suitable for listening tests. For example, a sentence containing a proper noun may be difficult to understand for someone who has never heard the word before, leading to a personal bias in listening tests. Meanwhile,

sentences that start with a preposition, particle, or conjunction, and sentences that end with a preposition or conjunction are also inappropriate because they are usually not complete sentences. Therefore, we used two pretrained POS tagging systems to tag candidate sentences and remove sentences that met the above POS-based removal criteria.

Perplexity is defined as the model's uncertainty regarding a sentence. Higher perplexity indicates that a sentence may be more difficult to understand. In this study, we used pre-trained BERT[7], a neural-network-based model trained with a masked language modeling objective, to compute the perplexity of each sentence. Given a sentence $W = (w_1, \ldots, w_i, \ldots, w_{|W|})$, w_i is the i-th character in W. To calculate W's perplexity, w_i is replaced with the [MASK] token and predicted using all other characters in W, that is, $W_{\setminus i} = (w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{|W|})$. $P_{BERT}(w_i|W_{\setminus i})$ is the probability of w_i given its context calculated by BERT. Then, the perplexity of sentence W is defined as:

$$Perplexity(W) = 2^{-\frac{1}{|W|}\sum_{i=1}^{|W|}\log_2 P_{BERT}(w_i|W_{\setminus i})} \quad (1)$$

A high $Perplexity(W)$ indicates that W contains characters that are difficult to predict from their context, suggesting that W can be difficult to understand. In this study, we keep $Perplexity(W)$ in the log space for ease of presentation. That is, we use $\text{PPL} = -\frac{1}{|W|}\sum_{i=1}^{|W|}\log P_{BERT}(w_i|W_{\setminus i})$ to represent the perplexity of sentence W. We computed the PPL for each sentence and analyzed the distribution of PPL across all sentences to determine a threshold. The perplexity filter then removes sentences whose PPL is above the threshold.

The last is the intelligibility filter, which removes sentences with low intelligibility scores. Figure 2 illustrates the calculation of the intelligibility score for a sentence. First, a TTS system was used to convert a sentence into a corresponding speech utterance. Subsequently, a pretrained automatic speech

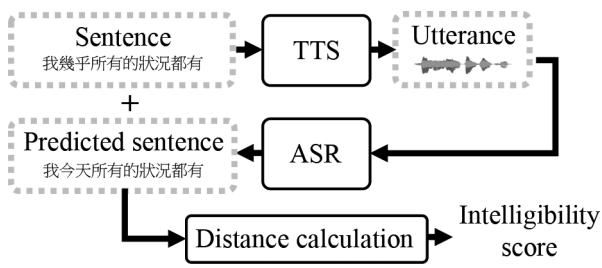

Figure 2: Illustration of the intelligibility score calculation. First, a sentence is converted into an utterance using a TTS system. Then, an ASR system is used to predict the content of the utterance. The distance between the sentence and ASR prediction is used to calculate the intelligibility score.

recognition (ASR) system is used to predict the content of the utterance. Finally, the Levenshtein distance between the sentence and the ASR prediction is used to measure the intelligibility of the sentence. If a sentence is difficult to understand, the TTS system may not be able to generate a correctly pronounced utterance because some characters have multiple pronunciations. In addition, previous research [6] showed that ASR predictions are highly correlated with human perception of intelligibility. In other words, if a sentence is confusing, the ASR system may fail to correctly recognize the corresponding speech utterance. Therefore, the distance between ASR prediction and the original sentence reflects the intelligibility of the sentence. The intelligibility score is defined as one minus the distance of the sentence divided by the length of the sentence. Therefore, a perfect ASR prediction will lead to an intelligibility score of 1.

2.2 Script-Composing

In the script-composing phase, we used the GA to select sentences, from the candidate sentences, to form a syllable-balanced recording script. The script consisted of several sets, each containing a fixed number of sentences, and the sentences did not overlap between sets. First, we introduce the basic concept of the GA. Then, we present the proposed GA-based script-composing method.

2.2.1 Genetic Algorithm (GA)

The GA is inspired by natural selection—a process of eliminating the weak and leaving only the strong. In the GA, the population is a series of possible solutions named chromosomes. Chromosomes are composed of genes that represent specific items. A fitness function is used to evaluate each chromosome. The fitness score reflects how well a chromosome "fits" the problem; a higher fitness score indicates that the chromosome is a better solution.

The GA comprises five steps: (1) initialization, (2) fitness calculation, (3) selection, (4) crossover, and (5) mutation. The initialization step creates the initial population and the fitness calculation step calculates the fitness score of each chromosome in the population. In the selection step, chromosomes with higher fitness scores have higher probabilities of leaving their offspring in the next generation. In the crossover step, a pair of selected chromosomes exchanges genes to form a new pair of chromosomes. Take one-point crossover as an example, a point called "crossover point" on both parents' chromosomes is randomly chosen. Then, the genes to the right of the crossover point are swapped between the parent chromosomes, producing two new chromosomes that carry genetic information from both parents. Lastly, genes in chromosomes may change randomly during the mutation step.

2.2.2 The GA-Based Script-Composing Phase

Figure 3 shows the GA terms and the corresponding definitions in this study. The *population* comprises a collection of scripts. Each *chromosome* is a script and the best chromosome in the population is the target syllable-balanced script. A *gene* is a sentence that is swapped between chromosomes.

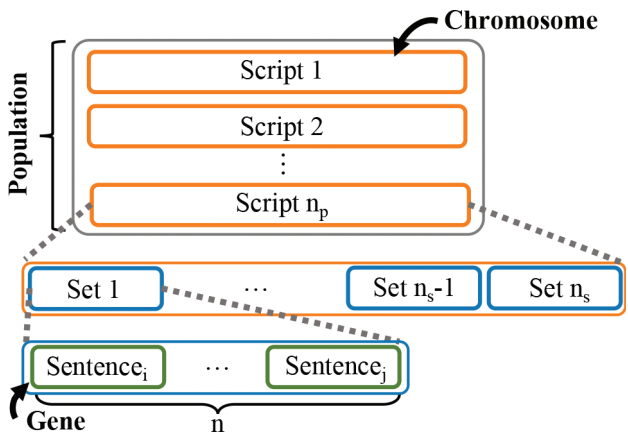

Figure 3: GA terms and their corresponding definitions. The *population* is a collection of scripts, each *chromosome* is a script, each *gene* is a sentence, and n_p, n_s, and n denote the number of scripts in the population, number of sets in a script, and number of sentences in a set, respectively. Sentence$_i$ denotes the i-th sentences in the candidate sentence set. The sentences were randomly sampled from the candidate sentence set during initialization, and there were no duplicate sentences in each script.

Figure 4 illustrates the GA process. The initial population step generated multiple scripts, each consisting of random sentences. The fitness calculation step then calculates the fitness score of each script in the population. The selection step replaces scripts with lower scores with scripts with higher fitness scores. The crossover step exchanges sentences between the scripts. This process stops when the population is dominated by one script and the maximum fitness score no longer increases. We skip the mutation step because it increases the complexity without improving the performance of our test.

2.2.3 Fitness Calculation

The fitness calculation step evaluates how well a script satisfies the requirements. Specifically, a script with a higher fitness score is considered a better choice.

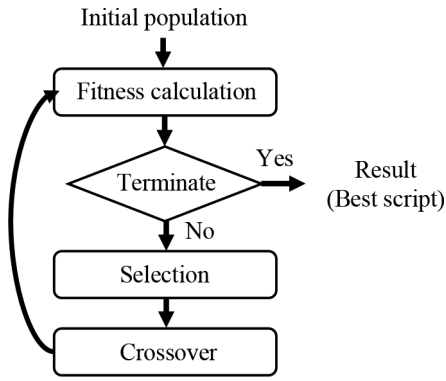

Figure 4: Schematic diagram of GA. The termination condition occurs when the maximum fitness score no longer increases after several generations.

In this study, the fitness score is defined as follows:

$$\begin{aligned} Fitness\_score = {} & w_1 \times script\_syllable\_distribution \\ & + w_2 \times script\_syllable\_coverage \\ & w_3 \times set\_syllable\_distribution, \end{aligned} \tag{2}$$

where w_1, w_2, and w_3 are the weights.

Let D_{script} be the syllable distribution of a script and D_{real} be the real-world syllable distribution, $D_{script} \in R^s$, $D_{real} \in R^s$, and s be the number of distinct syllables in Mandarin Chinese. The script_syllable_distribution is the cosine similarity between D_{script} and D_{real}.

$$script\_syllable\_distribution = \frac{D_{script} \cdot D_{real}}{\|D_{script}\| \, \|D_{real}\|}. \tag{3}$$

Similarly, the set_syllable_distribution is the average cosine similarity between the real-world syllable distribution and each set in the script.

$$set\_syllable\_distribution = \frac{1}{n_s} \sum_{i=1}^{n_s} \frac{D_{set}^i \cdot D_{real}}{\|D_{set}^i\| \, \|D_{real}\|}, \tag{4}$$

where D_{set}^i is the syllable distribution of the i-th set in the script, and n_s is the number of sets in the script. We include the set_syllable_distribution in the fitness score such that each set is representative and can be used individually. For example, each set can be used as a validation set in the training of a speech-processing model and as an indicator for selecting the best model. Additionally, each set can be used for model training when only a small amount of data is required.

Script_syllable_coverage is the fraction of all possible syllables covered in a script. For example, assuming that the number of distinct syllables in Mandarin Chinese is 1300, the script_syllable_coverage score of a script that contains 130 distinct syllables is 0.1 (i.e., 130/1300). Note that in this study, we consider *tonal* syllables instead of base syllables. In other words, the fitness function calculates the distribution and coverage of the tonal syllables.

2.2.4 Selection

The selection step realizes the "survival of the fittest." In other words, scripts with higher fitness scores are retained and replicated, whereas scripts with lower fitness scores are eliminated. In this study, the truncation selection method was used. Scripts were sorted by their fitness scores, and 50% of the fittest scripts were selected and replicated twice. Figure 5 shows the selection process.

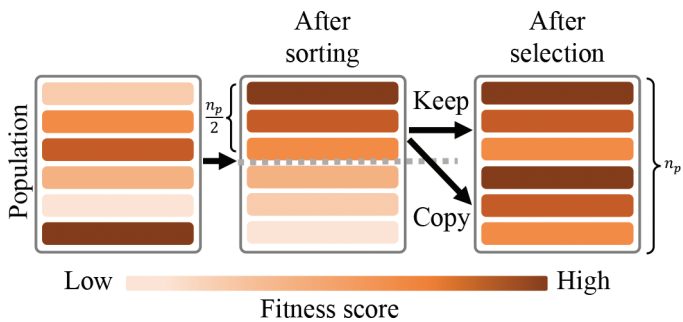

Figure 5: Illustration of the truncation-selection process. The scripts are sorted by their fitness scores, and then 50% of the fittest scripts are selected and replicated twice.

2.2.5 Crossover

The crossover step aims to combine the information of the two scripts and then generates new scripts. In this study, we used sets as crossover units, instead of complete scripts. This is because if we use scripts as crossover units, only one set in each script exchanges the information at every iteration when using the one-point crossover. However, if we use sets as crossover units, every set in the script participates in crossover at every iteration. Figure 6 shows an example of a crossover pair and Figure 7 illustrates the crossover step. As shown in Figure 7, to avoid duplicate sentences in one script, sentences present in the other script are held and not swapped in the crossover step. If the number of duplicate sentences in the paired sets is not the same, we randomly select

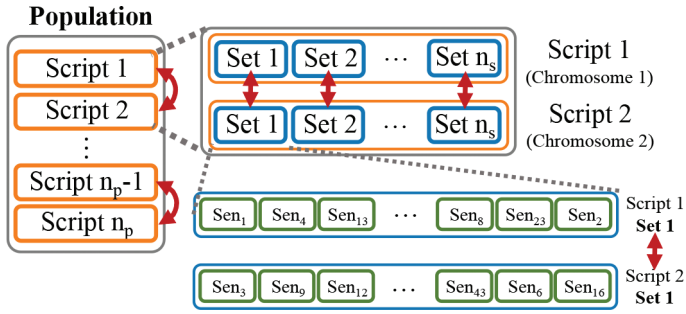

Figure 6: Illustration of the crossover pairs. The crossover step exchanges sentences between two sets with the same index.

sentences such that the number of held sentences is the same in both sets. Finally, we apply a one-point crossover to the two sets. Note that holding the same number of sentences in both sets ensures that the two new sets have the same number of sentences after crossover.

2.3 Postprocessing

After the script-composing phase, we obtained a syllable-balanced script. However, we may still want to replace some sentences in the script because the data-processing phase does not ensure that all candidate sentences are suitable. For example, the sensitive word filter cannot remove newly invented sensitive words that are not included in a sensitive word list. In addition, POS tagging systems may give incorrect POS tags because even the best POS tagging system cannot guarantee 100% accuracy. Therefore, sentences that meet POS removal criteria may not be removed as expected. Moreover, sentences with low PPLs and high intelligibility scores are not necessarily logical from the human perspective.

Therefore, in the postprocessing phase, we still need to manually label inappropriate sentences to be replaced with more appropriate sentences. The script generated in the script-composing phase is denoted as a temporary script. We propose two methods to replace unwanted sentences in a temporary script: (1) GA-based method and (2) greedy-based method.

The GA-based method is similar to the GA in the script-composing phase. The only difference is the generation of scripts in the initial population. In postprocessing, all scripts in the initial population are initialized based on the temporary script, with unwanted sentences replaced with sentences randomly sampled from the candidate sentences. The rest of the GA steps were the same as those in the script-composing phase. For the greedy-based method, unwanted sentences are replaced one by one with sentences from the candidate

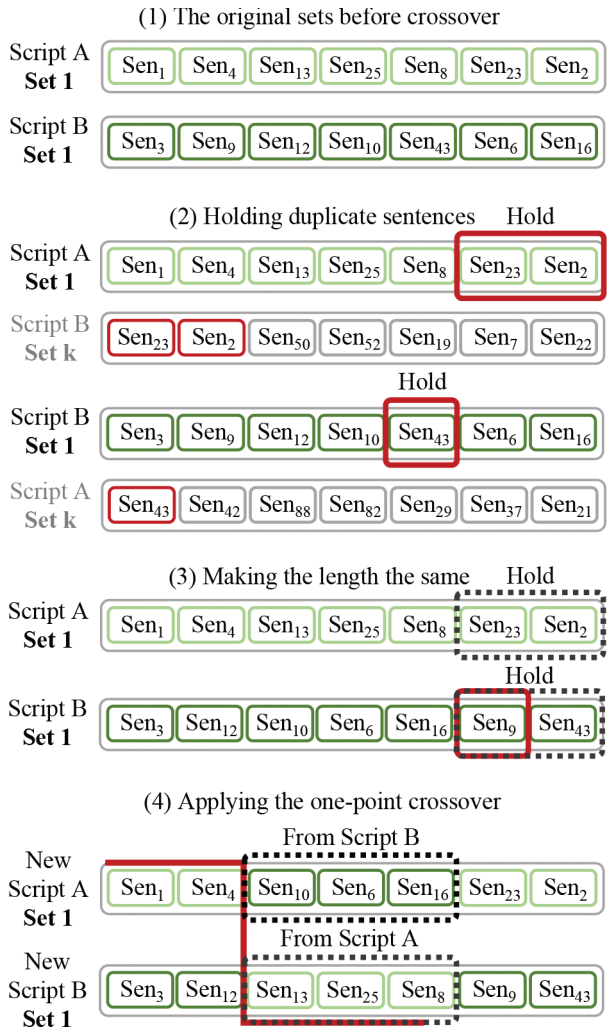

Figure 7: (1) The original sets before crossover. Sen_i denotes the i-th sentence in the candidate sentence set. (2) Holding duplicate sentences. In this example, Sen_{23} and Sen_2 are held because they exist in a set in Script B. Similarly, Sen_{43} is held because it already exists a set in Script A. These sentences are not exchanged during the crossover process to avoid duplicate sentences in the script. If Sen_{23} and Sen_2 are exchanged to Script B, there will be two Sen_{23} and two Sen_2 in Script B. (3) Making the length the same. Because the number of duplicate sentences in Set 1 of Script A and Set 1 of Script B are not the same (i.e., two sentences in Set 1 of Script A and one sentence in Set 1 of Script B), we randomly hold one more sentence (Sen_9) in Set 1 of Script B. (4) Applying the one-point crossover.

sentences that can achieve the highest fitness score. According to our empirical results, the greedy-based method is more suitable when there are only a few unwanted sentences in the temporary script, whereas the GA-based method is more suitable when there are many unwanted sentences in the script.

3 Experiments

In this section, we first present a statistical analysis of Mandarin speech units based on Chinese news articles collected from five major news media outlets in Taiwan in 2021. We then show that the proposed BASPRO system can effectively select sentences based on a specially designed fitness function to form a syllable-balanced script for collecting speech data. Finally, we demonstrate that speech processing models trained on a TTS-synthesized syllable-balanced speech corpus based on the syllable-balanced script can achieve better performance than their counterparts trained on a randomly composed speech corpus. Note that the "syllable distribution and coverage" in the experiments represent "*tonal* syllable distribution and coverage".

3.1 Analysis of News Articles in Taiwan in 2021

We crawled news articles from five major news media sources in Taiwan in 2021, with a total Chinese character count of around 182,583,000. We used the Pypinyin tool [20] to identify the syllables of each character. See Figure 13 and Table 14 in Appendix for the list of INITIAL and FINAL in the Pypinyin tool, and the INITIAL, FINAL, and tone distribution in these news articles. There are 404 distinct base syllables and 1259 distinct tonal syllables, which are close to the number of distinct base syllables and tonal syllables reported in other studies [30, 31]. Note that there is no consensus on the exact number of base and tonal syllables in Mandarin Chinese. For example, the number of base and tonal syllables in [30] are 416 and 1345, respectively, while in [31] they are 407 and 1333, respectively.

3.2 Data Processing Experiment

3.2.1 Experimental Settings of Data Processing

The general filter kept only ten-character sentences. The POS tagging filter removes sentences that satisfy the POS-based removal criteria using CkipTagger[14] or DDParser[36]. The removal criteria when using the CkipTagger and DDParser are listed in Table 1. The perplexity filter removes sentences with PPLs higher than 4.0. In intelligibility filter, only sentences with an intelligibility score of 1.0 were kept. After the data-processing phase, the total

Table 1: The POS-based Removal Criteria. Descriptions of POS tags can be found in [14] and [36].

Toolkit	Include	Start	End
CkipTagger[14]	'Nb','Nc','FW'	'DE','SHI','T'	'Caa','Cab','Cba', 'Cbb','P','T'
DDParser[36]	'LOC','ORG','TIME', 'PER','w','nz'	'p','u','c'	'xc','u'

Table 2: Data processing toolkits used in this study.

POS filter	Perplexity filter	Intelligibility filter	Syllable calculation
CkipTagger [14] DDParser [36]	Hugging Face [32] (bert-base-chinese)	Google-TTS [8] Google-ASR [34]	Pypinyin [20]

number of candidate sentences was around 167,000. Table 2 lists the toolkits used in each data-processing phase.

3.2.2 Experimental Results of Data Processing

Table 3 lists several examples of sentences and their corresponding PPLs. The experimental results showed that PPL can reflect human perception to a certain extent. Specifically, sentences 1-1, 2-1, and 3-1 are literally similar to sentences 1-2, 2-2, and 3-2, respectively. Only a few characters in each sentence pair were different, and the pronunciations of the different characters were similar. However, sentences 1-1, 2-1, 3-1 are considered natural, while sentences 1-2, 2-2, 3-2 contain typos or are illogical. According to the results in Table 3, sentences 1-2, 2-2, 3-2 have higher PPL, while sentences 1-1, 2-1, 3-1 have lower PPL. Figure 8 shows the PPL distribution for ten-character sentences in Mandarin Chinese news texts. The distribution of PPL was right-skewed, with a mean of 2.336. According to Figure 8, we chose 4.0 as the PPL threshold, which is approximately 1.5 standard deviations from the mean of PPL for all ten-character sentences. However, sometimes the PPL does not correctly reflect whether a sentence is understandable. For example, sentence 4 in Table 3 is difficult to understand but has the lowest PPL among the examples.

Table 4 lists examples of sentences and their corresponding intelligibility scores. "Ori" is the original input sentence, and "Pred" is the corresponding ASR prediction. The first and second examples show that the intelligibility filter can identify sentences with words that are not easy to understand. To avoid the need to replace many sentences in the postprocessing phase, the

Table 3: Examples of sentence PPL assessment.

Index	Content	Manual selection	PPL
1-1	他寧願當一匹孤獨的狼	✓	2.501
1-2	**那你願**當一**起**孤獨的狼	✗	**5.402**
2-1	警方就聞到他渾身酒味	✓	3.427
2-2	**喜歡**就聞到他**純**身酒味	✗	**6.091**
3-1	候選人也積極掃街拜票	✓	2.758
3-2	候選人也積極**少接待**票	✗	**5.913**
4	**達到與槓鈴跳舞的境界**	✗	**2.385**

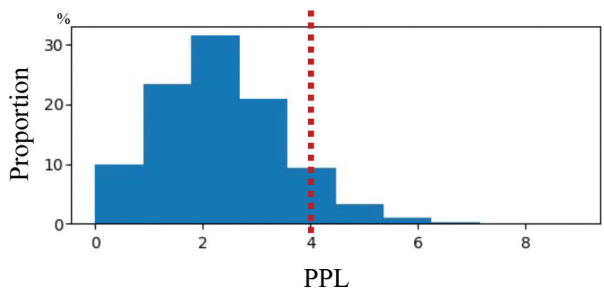

Figure 8: PPL distribution for ten-character sentences in Mandarin Chinese news texts. The red dotted line represents the threshold used for the perplexity filter.

intelligibility filter removes all sentences with an intelligibility score lower than 1.0. In other words, the intelligibility filter only retained sentences with perfect ASR test results. However, like PPL, sometimes, the intelligibility score does not perfectly reflect human perception. For example, the third sentence is not intuitive but has the intelligibility score of 1.0. As shown in Tables 3 and 4, perplexity and intelligibility filters cannot remove all illogical sentences. Therefore, manual labeling is required during the postprocessing phase.

3.3 GA-Based Script-Composing Experiment

In this section, we demonstrate that the BASPRO system can effectively select sentences to form a recording script according to the designed fitness function. We set the number of sets in the script and the number of sentences in each set to 20. Thus, the length of the chromosomes was 400. The weight of script_syllable_coverage (w_2 in Equation 2) was set to two, whereas the weights of the script_syllable_distribution (w_1 in Equation 2)

Table 4: Examples of sentence intelligibility assessment.

	Original sentence	Score	Comment
1-Ori	有一種**果敢**叫奮不顧身	0.8	The word "果敢" is rarely used in daily conversation.
1-Pred	有一種**果感**覺奮不顧身		
2-Ori	災害來臨時除了**盼天助**	0.7	The word "盼天助" is rarely used in daily conversation.
2-Pred	災害來臨時除了**看牽著**		
3-Ori & 3-Pred	**不科學的比例相當火辣**	**1.0**	**The sentence means "the unrealistic (body) proportions are very hot" in English. Because the sentence omits the subject "body," it is not intuitive and hard to understand.**

and set_syllable_distribution (w_3 in Equation 2) was set to 1. The population size was set to 25,000 and the GA was stopped until the maximum fitness score converged. Figure 9 shows the training curve of GA. The maximum fitness score drops for some generations because scripts are split and remixed in the crossover step, which may lower the fitness score. However, overall, the fitness score increases with the number of generations and eventually converges.

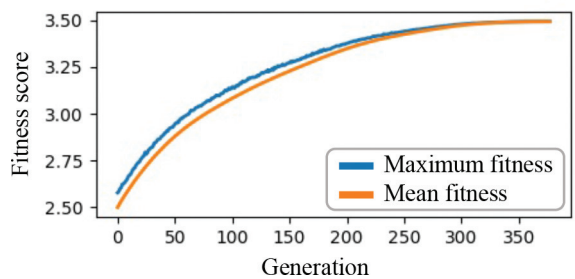

Figure 9: Training curve of the GA. Overall, the fitness score increases with the number of generations and then eventually converges.

Figure 10 shows the distribution of syllables in the best scripts of the first and final generations, and in real-world texts. The results showed that the syllable distribution of the best script in the final generation was much closer to the real-world syllable distribution than the syllable distribution of the best script in the first generation. The red region in Figure 10 indicates the effect of script_syllable_coverage score on the fitness function. In the real world, the ratio of the frequency of syllables with indices 800 to 1200 to the frequency of all syllables is close to 0; therefore, when considering only script_syllable_distribution and set_syllable_distribution in the fitness function, most syllables in this rare region will not be present in the best

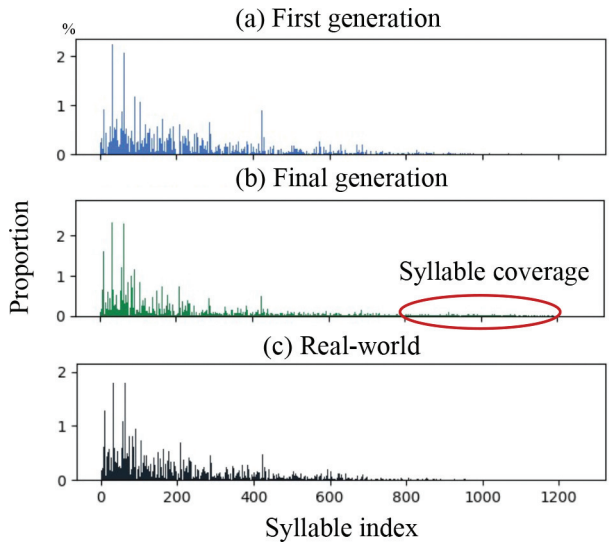

Figure 10: Distribution of syllables in the best scripts of the first and final generations and in real-world texts. The results show that the best script in the final generation has a syllable distribution that is closer to real-world syllable distribution than the best script in the first generation. The red region reveals the effect of script_syllable_coverage; that is, more rare syllables are covered in the best script in the final generation.

script in the final generation. However, because the fitness function includes script_syllable_coverage, more rare syllables are covered in the best script in the final generation, making the distribution of syllables indexed from 800 to 1200 in (b) and (c) significantly different.

Table 5 compares the values of script_syllable_distribution, set_syllable_distribution, and script_syllable_coverage for the best scripts in the first and final generations. Note that because there were 20 sets in a script, for the set_syllable_distribution, the mean and standard deviation of the 20 sets were calculated. Clearly, all values increase with generation. As shown in the ablation study in Table 6, there is a tradeoff between script_syllable_distribution, set_syllable_distribution, and syllable_coverage. For example, if the fitness function only considers the script_syllable_distribution, the best final script can achieve a script_syllable_distribution value of 0.997. However, in this case, the script_syllable_coverage and set_syllable_distribution can only reach 579 and 0.702, respectively.

Next, we compare the greedy and GA-based replacement methods in the postprocessing phase. Figure 11 shows the fitness scores of the resulting scripts for different replacement percentages. Specifically, 80% means that 320 (i.e., $400 \times 80\%$) sentences in the script have been replaced with new sentences.

Table 5: Statistics of the best scripts in the first and final generations.

Generation	Syllable coverage	Syllable distribution	
		Script	Set
First	668	0.894	0.622 (std: 0.033)
Final	1120	0.964	0.751 (std: 0.019)

Table 6: Ablation study of the fitness function.

Fitness function	Syllable coverage	Syllable distribution	
		Script	Set
All	1120	0.964	0.751(std:0.019)
Syllable coverage	**1122**	0.827	0.494(std:0.035)
Script distribution	579	**0.997**	0.702(std:0.042)
Set distribution	343	0.943	**0.889(std:0.003)**

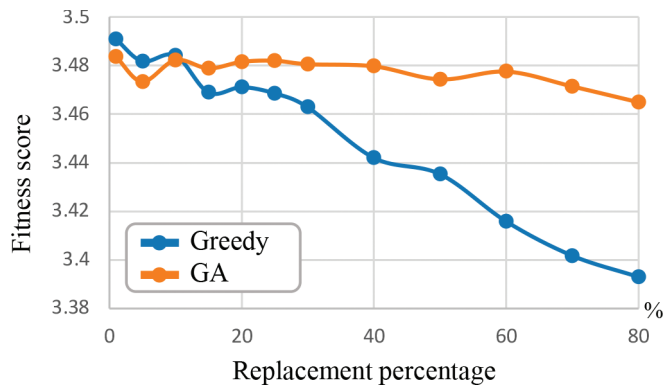

Figure 11: Comparison between the GA- and greedy-based replacement methods in the postprocessing phase. The greedy-based method outperforms the GA-based method when the replacement percentage is lower than 10%; however, as the replacement percentage increases, the GA-based method outperforms the greedy-based method.

The results show that if a large portion of sentences needs to be replaced, the GA-based method performs better than the greedy-based method. Conversely, if only a few sentences must be replaced, the greedy method outperforms the GA-based method.

Finally, Figure 12 compares the statistics of a script produced by the BASPRO system and the TMHINT Mandarin Chinese recording script [12] used in many previous studies. For a fair comparison, the number of sets and sentences in each set was set to 32 and 10, respectively, following the TMHINT script. The top two panels of Figure 12 show that the BASPRO-produced script covers more syllables, while the bottom two panels of Figure 12 show

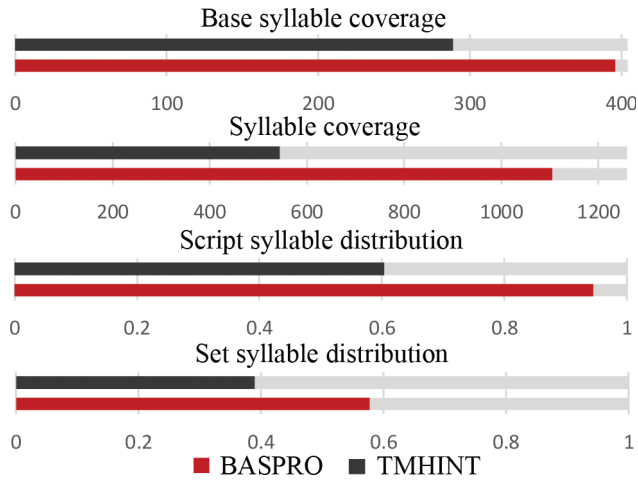

Figure 12: Comparison between the script produced by the BASPRO system and TMHINT script. The maximum base syllable and (tonal) syllable coverage were 404 and 1259, respectively. The maximum script and set syllable distribution scores are 1.

that the syllable distribution of the BASPRO-produced script is closer to the real-world syllable distribution.

3.4 Experiment on Speech-Processing Tasks

In this section, we investigate whether speech-processing models trained on the syllable-balanced *TMNews* corpus can outperform their counterparts trained on a randomly composed corpus. We experiment on two common speech processing tasks, including speech enhancement (SE) and ASR.

3.4.1 Experimental Settings for Both Tasks

To verify the usefulness of the proposed BASPRO system, we compared the performances of speech-processing models trained on syllable-balanced and randomly selected corpora. In the following experiments, CorpusBAL referred to a syllable-balanced corpus, whereas CorpusRAN represented a randomly composed corpus. CorpusBAL was formed based on a syllable-balanced script, *TMNews*. CorpusRAN was formed using randomly selected sentences. Both CorpusBAL and CorpusRAN have large and small versions, denoted by Corpus(BAL,RAN)_Large and Corpus(BAL,RAN)_Small, respectively. The large and small corpora contained 20 and 5 sets, respectively, with 20 sentences in each set. That is, 400 sentences form a large corpus and 100 sentences form a small corpus.

Table 7: Statistics of the speech corpora.

Corpus	Base syllable coverage	Syllable coverage	Script syllable distribution	Set syllable distribution
CorpusBAL_Large (TMNews_L)	392	1061	0.970	0.743 (std: 0.020)
CorpusBAL_Small (TMNews_S)	333	629	0.934	0.701 (std: 0.10)
CorpusRAN_Large	319	609	0.869	0.603 (std: 0.365)
CorpusRAN_Small	241	387	0.818	0.637 (std: 0.015)

For each sentence in the script, we used two TTS systems, GoogleTTS [8] and TTSkit [26], to generate corresponding utterances. The utterances generated by GoogleTTS were female voices, while the utterances generated by TTSkit were male voices. As a result, the Corpus(BAL,RAN)_Large corpus contains 800 utterances, and the Corpus(BAL,RAN)_Small corpus contains 200 utterances.

Table 7 lists the statistics of each speech corpus. The syllable distribution of CorpusBAL was closer to the real-world syllable distribution than that of CorpusRAN. In addition, CorpusBAL_Small had better syllable coverage than CorpusRAN_Large, although the number of sentences in CorpusBAL_Small was only a quarter of that in CorpusRAN_Large.

Experimental Settings for the SE Task. We trained the SE model on small corpora, and tested it on large corpora. In practical applications, the test data are also larger than the training data. Therefore, we believe that the experimental results under this setting can better reflect performance in a real environment.

For the training data, each clean utterance was contaminated with 25 noises randomly selected from 100 noises [11] at −1, 1, 3, and 5 SNR levels. The training data contained 20,000 utterances (100 (sentences) × 2 (voice types) × 25 (noise types) × 4 (SNR levels)). The training data were divided into training and validation datasets. The validation set contained 20% of the training data and was used to select the best model for training. Therefore, in our experiments, using this training–validation setup, we trained five models with a training corpus and reported the mean and standard deviation of the results evaluated on the testing corpus. For the test set, each clean utterance was contaminated with three noise types (white, street, and babble) at 2 and 4

SNR levels. The test set contains 4800 utterances (400 (sentences) × 2 (voice types) × 3 (noise types) × 2 (SNR levels)).

The corpora were evaluated using MetricGAN+[9, 10], a state-of-the-art SE model. Because the input of MetricGAN+ is a spectrogram, the input signal was transformed into a spectrogram using a short-time Fourier transform (STFT) with a window length of 512 and hop length of 256. In addition, the batch size was 32, the loss function used was L1 loss, and the optimizer was Adam with a learning rate of 0.001. The perceptual evaluation of speech quality (PESQ) [23] and short-time objective intelligibility (STOI) [25] are used as objective evaluation metrics.

Experimental Settings for the ASR Task. In the ASR experiments, we downloaded the pretrained transformer-based ASR model from SpeechBrain [21], and then fine-tuned the ASR model using the speech corpora collected in this study. The pre-trained ASR model was trained on the AISHELL dataset, which is also a Mandarin speech corpus. We fine-tuned a pre-trained model because our training speech was not sufficient to train the ASR model from scratch. In addition, this setup simulates the personalization of an ASR system, that is, fine-tuning an ASR system with a few recordings of a new user. Similar to the 80% training-20% validation setting in the SE task, given a training corpus, we obtained five models and reported the means and standard deviations of the evaluation results. For each training and validation split, we fine-tuned the model for 50 epochs and selected the best model using a validation set.

We used the pinyin error rate (PER), character error rate (CER), and sentence error rate (SER) to evaluate ASR performance. PER calculates the difference between the predicted and ground-truth syllable sequences. Note that Pypinyin [20] was used to convert characters to tonal syllables before calculating PER. PER and CER were calculated using Levenshtein distance. In SER, a predicted sentence is considered to be incorrect if any character is wrong.

3.4.2 Experimental Results for SE

Table 8 compares the performances of the SE models trained on CorpusBAL_Small and CorpusRAN_Small. The results show that the SE model trained on CorpusBAL_Small outperformed the SE model trained on CorpusRAN_Small in terms of both PESQ and STOI under all testing conditions. In addition, both models performed worse when tested on CorpusBAL_Large than on CorpusRAN_Large. This may be because CorpusBAL_Large covers more syllables than CorpusRAN_Large, thus making it a more challenging test corpus. Table 9 presents the corresponding t-test results. The p-values of the STOI results on both CorpusBAL_Large and CorpusRAN_Large testing

Table 8: Performance of the SE models trained on CorpusBAL and CorpusRAN.

Testing \ Training	CorpusBAL_Small		CorpusRAN_Small	
	STOI	PESQ	STOI	PESQ
CorpusBAL_Large	**0.832** (std: 0.0149)	**1.792** (std: 0.1154)	0.793 (std: 0.0426)	1.744 (std: 0.1068)
CorpusRAN_Large	**0.832** (std: 0.0133)	**1.804** (std: 0.1182)	0.796 (std: 0.0426)	1.755 (std: 0.1101)

Table 9: T-test of the CorpusBAL_Small and CorpusRAN_Small SE results.

	p-value	
Testing data	STOI	PESQ
CorpusBAL_Large	0.10028	0.51936
CorpusRAN_Large	0.10524	0.46861

data are about 0.1, while the p-values of the PESQ results are about 0.5. That is, the improvement in the SE performance on STOI is more statistically significant than that on PESQ. This result may be because syllable coverage and distribution have a greater impact on intelligibility (STOI) than on quality (PESQ).

The fitness function contains the set_syllable_distribution score, because we want each set to be representative. We argue that the model selected by a small syllable-balanced validation set is more robust than the model selected by a small randomly selected validation set. Table 10 compares the performances of the SE models selected with different validation sets. The SE model was trained on CorpusBAL_Small and tested on CorpusBAL_Large and CorpusRAN_Large. In Table 10, valid:bal indicates that the validation set is a syllable-balanced set in CorpusBAL_Small, whereas valid:ran indicates that the validation set is randomly selected sentences from CorpusBAL_Small. The results show that the average performance of the SE models selected with a syllable-balanced validation set is better than that of the SE models selected with a randomly selected validation set.

Table 10: SE performance using different validation sets.

Testing \ Training	CorpusBAL_Small (valid:bal)		CorpusBAL_Small (valid:ran)	
	STOI	PESQ	STOI	PESQ
CorpusBAL_Large	**0.832** (std: 0.0149)	**1.792** (std: 0.1154)	0.814 (std: 0.0266)	1.790 (std: 0.0655)
CorpusRAN_Large	**0.832** (std: 0.0133)	**1.804** (std: 0.1182)	0.816 (std: 0.0265)	1.802 (std: 0.0694)

3.4.3 Experimental Results for ASR

Table 11 shows the performance of the ASR models fine-tuned using CorpusBAL and CorpusRAN. First, the results reveal that fine-tuning an ASR model always improves ASR performance. In addition, the ASR models fine-tuned on CorpusBAL generally performed better than their corresponding models fine-tuned on CorpusRAN. This is because the CorpusBAL_Large and CorpusBAL_Small corpora cover relatively complete and rich pronunciations; thus, the ASR model can be fine-tuned comprehensively. However, we also see that when tested on CorpusRAN_Small, the ASR model fine-tuned on CorpusBAL_Large performs slightly worse than the ASR model fine-tuned on CorpusRAN_Large. One possible explanation is that both CorpusBAL_Large and CorpusRAN_Large cover more syllables than CorpusRAN_Small, as shown in Table 7. Therefore, fine-tuning the model with either corpus did not make a significant difference when testing on a small test set. However, such a biased small test set could mislead the model. When using a small corpus as a test set, more consideration should be given to the pronunciation balance and coverage. Finally, the ASR performance tested on CorpusRAN is better than the ASR performance tested on CorpusBAL, which is consistent with the SE experiments. This is because CorpusBAL covers more rare syllables and is, therefore, more challenging than CorpusRAN.

Table 11: Performance of ASR models trained on CorpusBAL and CorpusRAN.

Testing data	Training data	PER	CER	SER
CorpusBAL_Large	w/o fine-tuned	14.94	19.73	74.88
	CorpusBAL_Small	**9.658** (std: 0.259)	**15.544** (std: 0.212)	**67.648** (std: 0.957)
	CorpusRAN_Small	10.738 (std: 0.277)	16.696 (std: 0.264)	70.324 (std: 1.311)
CorpusRAN_Large	w/o fine-tuned	8.78	11.69	55.62
	CorpusBAL_Small	**4.885** (std: 0.062)	9.244 (std: 0.141)	**47.922** (std: 0.518)
	CorpusRAN_Small	5.063 (std: 0.034)	**9.094** (std: 0.186)	49.126 (std: 0.905)
CorpusBAL_Small	w/o fine-tuned	14.30	17.75	70.00
	CorpusBAL_Large	**6.61** (std: 0.163)	**11.84** (std: 0.397)	**56.70** (std: 1.823)
	CorpusRAN_Large	7.91 (std: 0.357)	13.08 (std: 0.529)	61.30 (std: 3.114)
CorpusRAN_Small	w/o fine-tuned	8.55	12.30	53.50
	CorpusBAL_Large	3.24 (std: 0.221)	7.89 (std: 0.433)	**42.20** (std: 1.483)
	CorpusRAN_Large	**3.02** (std: 0.103)	**7.41** (std: 0.379)	44.20 (std: 1.483)

Table 12 presents the corresponding t-test results. This evaluation shows that the performance of the two ASR models using corpora of different scripts across all evaluation metrics is significantly different on the CorpusBAL_Large and CorpusBAL_Small testing data (p-value $\ll 0.05$). On the CorpusRAN_Large testing data, the p-value for CER is 0.18967, which means that the performance difference is not significant. Note that the CER is the only case in which CorpusRAN_Small performs better than CorpusBAL_Small on CorpusRAN_Large in Table 11. On the CorpusRAN_Small testing data, the performance differences in PER, CER, and SER are not significant (p-value$>$0.05). The experimental results show that syllable coverage and distribution should be considered for both training data and testing data, especially when the amount of data is small.

Table 12: T-test of the CorpusBAL and CorpusRAN ASR results.

	p-value		
Testing data	PER	CER	SER
CorpusBAL_Large	0.00022	0.00006	0.00617
CorpusRAN_Large	0.00051	0.18967	0.03256
CorpusBAL_Small	0.00008	0.00305	0.02148
CorpusRAN_Small	0.07949	0.09961	0.06559

Table 13 compares the performance of best model selection using different validation sets. The ASR model was fine-tuned on CorpusBAL_Small and tested on CorpusBAL_Large and CorpusRAN_Large. The best model was selected using a syllable-balanced set (cf. valid:bal in Table 13) or a randomly selected sentences set (cf. valid:ran in Table 13). The results show that the ASR model selected by a syllable-balanced validation set yields lower CER and SER than the ASR model selected by a randomly selected validation set.

Table 13: ASR performance using different validation sets.

Testing data	Training data	PER	CER	SER
CorpusBAL_Large	CorpusBAL_Small (valid:bal)	9.658 (std: 0.259)	**15.544** (std: 0.212)	**67.648** (std: 0.957)
	CorpusBAL_Small (valid:ran)	**9.630** (std: 0.263)	15.622 (std: 0.274)	67.898 (std: 0.672)
CorpusRAN_Large	CorpusBAL_Small (valid:bal)	4.885 (std: 0.062)	**9.244** (std: 0.141)	**47.922** (std: 0.518)
	CorpusBAL_Small (valid:ran)	**4.870** (std: 0.132)	9.250 (std: 0.168)	47.976 (std: 0.445)

4 Conclusion

In this paper, we first present a statistical analysis of Mandarin Chinese acoustic units based on a large corpus of news texts collected from the internet. We then proposed the BASPRO system that selects sentences from a large text corpus to compose a syllable-balanced recording script with similar statistics. The experimental results showed that the BASPRO system can effectively produce a syllable-balanced script based on the designed fitness function. Using BASPRO, we obtained a recording script called *TMNews*. Subsequently, we used TTS systems to convert sentences in the *TMNews* script into utterances to form a speech corpus. Through SE and ASR experiments evaluated on speech corpora based on different recording scripts, we confirmed that SE and ASR models trained on a syllable-balanced speech corpus based on the *TMNews* script outperformed those trained on a randomly formed speech corpus. In this study, we primarily focused on the design of audio-recording scripts rather than the audio recordings. There are too many variations in the recorded utterances, such as the recording device and the gender, age, and accent of the speaker. Therefore, the recording setting is beyond the scope of this study, and we used synthetic speech with relatively simple characteristics for the SE and ASR evaluation experiments. Furthermore, the data-processing phase does not ensure that every candidate sentence is logical and appropriate from a human perspective. Therefore, manual screening is required during the postprocessing phase. In the future, we hope to develop a method that better reflects human understanding of sentence semantics and reduces human involvement in corpus design.

Appendix

Table 14: The INITIAL and FINAL list in the Pypinyin tool.

(a) INITIAL list

Bopomofo	ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ	ㄐ	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ
Pinyin	b	p	m	f	d	t	n	l	g	k	h	j	q	x	zh	ch	sh	r	z	c	s
Example	不	頗	摸	費	得	特	那	樂	歌	科	喝	幾	七	西	之	吃	師	日	茲	雌	斯

(b) FINAL list

er 儿 兒	a ㄚ 啊	o ㄛ 喔	e ㄜ 鵝	ai ㄞ 哎	ei ㄟ 欸	ao ㄠ 熬	ou ㄡ 歐	an ㄢ 安	en ㄣ 森	ang ㄤ 昂	eng ㄥ 亨	ong ㄨㄥ轟
i ㄧ 衣	ia ㄧㄚ呀	io ㄧㄛ唷	ie ㄧㄝ耶			iao ㄧㄠ腰	iou ㄧㄡ優	ian ㄧㄢ煙	in ㄧㄣ因	iang ㄧㄤ央	ing ㄧㄥ英	iong ㄩㄥ雍
u ㄨ 烏	ua ㄨㄚ哇	uo ㄨㄛ窩		uai ㄨㄞ歪	uei ㄨㄟ威			uan ㄨㄢ灣	uen ㄨㄣ溫	uang ㄨㄤ汪	ueng ㄨㄥ翁	
v ㄩ 迂			ve ㄩㄝ約					van ㄩㄢ淵	vn ㄩㄣ暈			

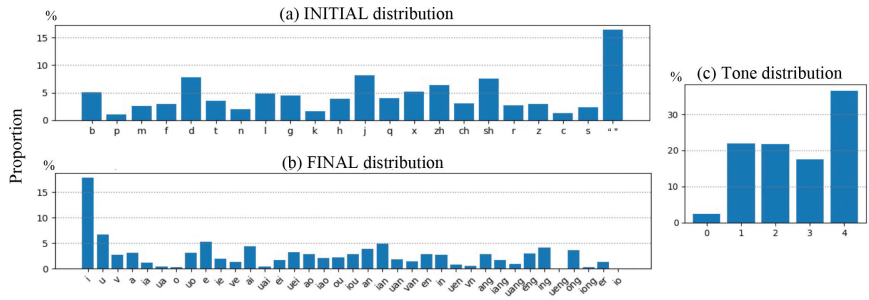

Figure 13: The INITIAL, FINAL, and tone distribution in news articles crawled from five major news media in Taiwan in 2021. " " in (a) refers to the syllable pronunciation without INITIAL.

References

[1] S. T. Abate, W. Menzel, B. Tafila, *et al.*, "An Amharic speech corpus for large vocabulary continuous speech recognition," in *Proc. INTERSPEECH 2005*.

[2] M. A. Abushariah, R. N. Ainon, R. Zainuddin, M. Elshafei, and O. O. Khalifa, "Phonetically rich and balanced text and speech corpora for Arabic language," *Language Resources and Evaluation*, 46(4), 2012, 601–34.

[3] A. Ahmad, M. R. Selim, M. Z. Iqbal, and M. S. Rahman, "SUST TTS Corpus: a phonetically-balanced corpus for Bangla text-to-speech synthesis," *Acoustical Science and Technology*, 42(6), 2021, 326–32.

[4] M. A. Bashar and R. Nayak, "Active learning for effectively fine-tuning transfer learning to downstream task," *ACM Transactions on Intelligent Systems and Technology*, 12(2), 2021, 1–24.

[5] B. Bozkurt, O. Ozturk, and T. Dutoit, "Text design for TTS speech corpus building using a modified greedy selection," in *Proc. Eurospeech 2003*.

[6] Y.-W. Chen and Y. Tsao, "InQSS: a speech intelligibility and quality assessment model using a multi-task learning network," in *Proc. INTERSPEECH 2022*.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL 2019*.

[8] P.-N. Durette, "Google Text-to-Speech," 2022 (accessed on August 01, 2022), https://pypi.org/project/gTTS/.

[9] S.-W. Fu, C.-F. Liao, Y. Tsao, and S.-D. Lin, "MetricGAN: generative adversarial networks based black-box metric scores optimization for speech enhancement," in *Proc. ICML 2019*.

[10] S.-W. Fu, C. Yu, T.-A. Hsieh, P. Plantinga, M. Ravanelli, X. Lu, and Y. Tsao, "Metricgan+: An improved version of metricgan for speech enhancement," in *Proc. INTERSPEECH 2021*.

[11] G. Hu and D. Wang, "A tandem algorithm for pitch estimation and voiced speech segregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 18(8), 2010, 2067–79.

[12] M. Huang, "Development of Taiwan Mandarin hearing in noise test," *Department of Speech Language Pathology and Audiology, National Taipei University of Nursing and Health Science*, 2005.

[13] A. Kurematsu, K. Takeda, Y. Sagisaka, S. Katagiri, H. Kuwabara, and K. Shikano, "ATR Japanese speech database as a tool of speech recognition and synthesis," *Speech Communication*, 9(4), 1990, 357–63.

[14] P.-H. Li, T.-J. Fu, and W.-Y. Ma, "Why attention? analyze BiLSTM deficiency and its remedies in the case of NER," in *Proc. AAAI 2020*.

[15] M.-s. Liang, R.-y. Lyu, and Y.-c. Chiang, "An efficient algorithm to select phonetically balanced scripts for constructing a speech corpus," in *Proc. NLP-KE 2003*.

[16] J. Luo, J. Wang, N. Cheng, and J. Xiao, "Loss prediction: End-to-end active learning approach for speech recognition," in *Proc. IJCNN 2021*.

[17] M. V. Nicodem, I. C. Seara, R. Seara, and D. dos Anjos, "Recording script design for a Brazilian Portuguese TTS system aiming at a higher phonetic and prosodic variability," in *Proc. ISSPA 2007*.

[18] M. V. Nicodem, I. C. Seara, D. d. Anjos, and R. Seara, "Evolutionary-based design of a Brazilian Portuguese recording script for a concatenative synthesis system," in *Proc. PROPOR 2008*.

[19] Y. R. Oh, Y. G. Kim, M. Kim, H. K. Kim, M. S. Lee, and H. J. Bae, "Phonetically balanced text corpus design using a similarity measure for a stereo super-wideband speech database," *IEICE Transactions on Information and Systems*, 94(7), 2011, 1459–66.

[20] "Pypinyin," 2022 (accessed on August 01, 2022), https://github.com/mozillazg/python-pinyin.

[21] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "SpeechBrain: A general-purpose speech toolkit," arXiv:2106.04624, 2021, arXiv: 2106.04624 [eess.AS].

[22] A. A. Raza, S. Hussain, H. Sarfraz, I. Ullah, and Z. Sarfraz, "Design and development of phonetically rich Urdu speech corpus," in *Proc. O-COCOSDA 2009*.

[23] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *Proc. ICASSP 2001*.

[24] M. Stănescu, H. Cucu, A. Buzo, and C. Burileanu, "ASR for low-resourced languages: building a phonetically balanced Romanian speech corpus," in *Proc. EUSIPCO 2012*.

[25] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7), 2011, 2125–36.

[26] "Text To Speech Toolkit," 2022 (accessed on August 01, 2022), https://pypi.org/project/ttskit/.

[27] K.-S. Tsai, L.-H. Tseng, C.-J. Wu, and S.-T. Young, "Development of a Mandarin monosyllable recognition test," *Ear and Hearing*, 30(1), 2009, 90–9.

[28] E. Uraga and C. Gamboa, "VOXMEX speech database: design of a phonetically balanced corpus," in *Proc. LREC 2004*.

[29] L. Villaseñor-Pineda, M. Montes-y-Gómez, D. Vaufreydaz, and J.-F. Serignat, "Experiments on the construction of a phonetically balanced corpus from the web," in *Proc. CICLing 2004*.

[30] H.-m. Wang, "Statistical analysis of Mandarin acoustic units and automatic extraction of phonetically rich sentences based upon a very large Chinese text corpus," 3(2), 1998, 93–114.

[31] H.-M. Wang, Y.-C. Chang, and L.-S. Lee, "Automatic selection of phonetically rich sentences from a Chinese text corpus," in *Proc. ROCLING 1993*.

[32] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, "Huggingface's transformers: state-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[33] C. Wutiwiwatchai, P. Cotsomrong, S. Suebvisai, and S. Kanokphara, "Phonetically distributed continuous speech corpus for Thai language," in *Proc. LREC 2002*.

[34] A. Zhang, "Speech Recognition (version 3.8)," 2017 (accessed on August 01, 2022), https://github.com/Uberi/speech_recognition#readme.

[35] J. T. F. L. M. Zhang and H. Jia, "Design of speech corpus for Mandarin text to speech," in *Proc. The Blizzard Challenge 2008 Workshop*.

[36] S. Zhang, L. Wang, K. Sun, and X. Xiao, "A practical Chinese dependency parser based on a Large-scale dataset," 2020, arXiv: 2009.00901 [cs.CL].

[37] V. Zue, S. Seneff, and J. Glass, "Speech database development at MIT: TIMIT and beyond," *Speech Communication*, 9(4), 1990, 351–6.