***Information system for developing listening comprehension skills in foreign language learning***

Lytovchenko Alina, Zhurakovska Oksana

Department of Information Systems and Technologies

Igor Sikorsky Kyiv Polytechnic Institute

Kyiv, Ukraine

alina.lytovchenko20@gmail.com, [o.zhurakovska@kpi.ua](o.zhurakovska%40kpi.ua)

*Abstract.* This paper presents an intelligent system for improving listening comprehension in foreign language learning. It converts user-uploaded English texts into synchronized audio and subtitles, with instant translation support. To reduce delays, a dynamic programming algorithm optimizes parallel TTS requests. Experiments show that the method significantly speeds up audio generation, aiding especially Ukrainian users adapting to new language environments.

Keywords: Text-to-Speech, Interactive Learning, Speech Synthesis, Artificial Intelligence, Automatic Translation, Listening Skills, Audiobook Generation, Dynamic Programing Method.

# Introduction

In the era of rapid development of information technology, the ability to effectively develop foreign language listening skills is becoming important for both personal and professional development. Listening is one of the most complex competencies, the development of which requires regular training, considerable time and high-quality educational resources. The introduction of artificial intelligence methods into the learning process can increase its effectiveness, including by creating adaptive and interactive audio content that meets the needs of each user.

The relevance of this work is determined by the acute need of Ukrainians for effective means of language adaptation, especially in connection with forced relocation due to military operations. The developed information system ensures the organization of the development of listening skills in the learning process by automatically creating high-quality audio accompaniment to textual materials in English, synchronizing audio with the visual display of the text, and an integrated translation tool, which greatly facilitates learning and helps to overcome the language barrier.

## Structure of the Information System for Developing Listening Comprehension Skills in Foreign Language Learning

 The field of research of this paper is the process of developing listening skills in learning a foreign language and methods of its improvement. The developed system provides automatic generation of audio accompaniment to any English-language texts uploaded by users, provides synchronous text highlighting while listening to it, and allows instant translation of any selected text fragments.

 The structural diagram of the information system illustrates the interaction between its main components and the technologies that ensure its full functionality. The structural diagram is shown in Figure 1.



 Fig. 1 - Structural Diagram of the Information System for Developing Listening Comprehension Skills in Foreign Language Learning

At the first stage, the user interacts with the system via a client device (computer, smartphone, or tablet). The user's device runs a web browser that provides access to the system's web interface.

The system interface is implemented using React technologies and the Next.js framework, which form the front-end application. This component is responsible for presenting information and transmitting user requests to the backend server.

The backend part of the system is also built on the basis of Next.js and processes all received requests by interacting with the following components:

* AWS S3 cloud file storage that stores book files, generated audio files, subtitles (SRT files), and book covers;
* a Flask-based Python service responsible for generating audio files for text materials. This service accesses the OpenAI TTS API, which provides high-quality and natural text-to-speech;
* a cloud-based PostgreSQL Neon database that stores information about users, books, reading progress, and other necessary metadata;
* DeepL API as an external translation service used to instantly translate text fragments from English into Ukrainian in real time.

 This architecture ensures stable, scalable and efficient operation of the developed information system.

*Problem Statement*

 The goal of the development is to ensure minimal delay in generating the voiceover of the English text uploaded by the user. An important requirement for the system's operation is the almost instantaneous start of audio playback and the receipt of synchronized subtitles immediately after the text file is uploaded. For this purpose, the Text-to-Speech technology is used with the use of streaming text processing in parts. It is planned to use the OpenAI speech synthesis API, which supports the transmission of audio in fragments, allowing you to start playback before the processing of the entire text is complete.

The task is formulated as follows:

— the user uploads the text of a book consisting of a sequence of paragraphs;

— the text is divided into a set of fragments, each of which corresponds to one paragraph;

— fragments are grouped into packages, each package containing one or more paragraphs;

— the packets are processed sequentially, and the fragments within each packet are processed in parallel through parallel requests to the TTS API. The number of simultaneous parallel requests within a single packet is limited by a set limit;

— the generation time of audio and subtitles for each package is defined as the maximum generation time of one paragraph included in the package;

— the total generation time is equal to the sum of the generation time for all packets.

The optimization task is to find a grouping of paragraphs into batches that minimizes the total time for generating audio and subtitles. The packages should not have any common paragraphs.

 The criterion for system efficiency is the minimum time from the moment the text is loaded to the start of audio playback and the total time for generating the entire text.

 The proposed solution will allow the user to start listening to the book almost immediately after downloading it, while simultaneously receiving accurate audio in MP3 format and synchronized subtitles in SRT format.

 Formally, the input data of the problem is defined as follows:

 The input data of the problem are:

 ***T*** – the text of the book downloaded by the user;

 – i-th paragraph of the book, , , where *N* = |*T*|;

 P– the maximum number of simultaneous requests;

 ***F*** – the set of fragments formed from *T* after splitting the book into paragraphs: ***F*** = {,, …,}, |*F*| ≤ *P*;

 *G* – the set of packages for parallel processing:

 *G*= {,,…,}, where each⊂ *F*, , *K* – amount of packages;

 ***S*** – a set of sequential requests to the TTS API:

 ***S***= {,,…,};

 *t*($)$ - the time of audio and subtitle generation for the packet, = max (t ());

 – total generation time, =$.$

 **Thus, the mathematical formulation of the problem is to find a structure for splitting the set *F* into *K* packets (where *K* ≤ *P*) with the number of fragments in each packet |****|, minimizing the value of** **, i.e:**

=→ **min,**

under the conditions of: ⋃ = ***F***, ∩ = ∅ at *k* ≠ l.

 The proposed solution will allow the user to start listening to the book almost immediately after downloading it, while receiving accurate audio in MP3 format and synchronized subtitles in SRT format.

*Justification of the Solution Method*

The easiest way to solve this problem is to make consecutive requests to the OpenAI TTS API, where each request returns audio for text no longer than 4096 characters (the maximum limit of one request). However, this approach can be quite time-consuming, so there is a need to find a more efficient solution.

 In modern research, various approaches are widely used to solve optimization problems: evolutionary algorithms [6], multi-agent methods (particle swarm algorithms [5], ant colonies [4]), multi-criteria decision-making methods [3], and dynamic programming algorithms [2].

 Evolutionary algorithms imitate the processes of natural evolution and genetics, allowing to obtain approximate solutions to complex problems. Their main advantage is flexibility in customization, but they require complex parameter calibration [6].

 Optimization by ant colonies and particle swarms is based on modeling the behavior of natural systems and is effective in large search spaces. The disadvantages of these methods are high computational costs and complexity of setup [4].

 Multi-criteria decision-making methods are used when a task has several optimization criteria, which allows finding compromise solutions, but makes it difficult to find the single best option [3].

 Dynamic programming (DP) algorithms solve a problem by breaking it down into smaller subproblems for which optimal solutions are found. They are particularly effective in problems that have an optimal substructure and overlapping subproblems. These characteristics make a DP the best choice for our task, which is to split text into packets while minimizing the time to generate audio [2].

 Given this, the dynamic programming algorithm was chosen to solve the problem because of its ability:

* consistently and in detail consider all possible options for splitting paragraphs into packages without missing any potentially optimal solutions;
* to find the exact solution without additional complex settings;
* is easy to adapt to the real limitations of parallel processing.

To solve the problem, we use a dynamic programming algorithm that consists of the following steps:

 Step 1. Formation of the input data structure.

 At the initial stage, a list of paragraphs is generated. For each paragraph, a pre-calculated synthesis time is determined.

 Also, the parallel processing limit *k* is determined - the maximum number of paragraphs that can be simultaneously synthesized within one package (corresponds to the number of simultaneous requests to the TTS API).

 Step 2. Splitting into subtasks using the dynamic programming method.

 The algorithm checks different ways of splitting the text into packets and looks for the one with the lowest total time. To do this, we introduce the variable *dp*[*i*], which means “the minimum possible time to read the first i paragraphs of the book.”

 To find *dp*[*i*], all possible options for building the latest package are considered (for example, consisting of one paragraph  or 2 paragraphs and , and so on). Then, we determine the voice time of this package, which is added to the voice time of the previous paragraphs. Choose the minimum result among all the options.

 Step 3. Calculating the final result.

 After sequentially calculating *dp*[1], *dp*[2],..., *dp*[*n*] the last value obtained *dp*[*n*] will be the minimum sounding time for the entire book.

*Analysis of the Obtained Results*

 Let's conduct a series of experiments to compare the speed of the basic algorithm, in which audio is generated sequentially (request after request, each containing no more than 4096 characters), with the proposed method, which uses the optimal distribution of text into packets for parallel audio generation.

 A comparison of the results is shown in the graph in Figure 2.



Fig. 2 - Performance Comparison of Baseline and Optimized (Parallel) TTS Algorithms

 The analysis of the results allows us to conclude that the proposed optimized method is significantly superior to the baseline method, where audio is generated sequentially, request by request every 4096 characters. The baseline method shows a significant increase in generation time with increasing text size, as each new packet waits for the previous one to complete.

 The proposed method, which uses a dynamic programming algorithm to efficiently divide the text into packets and process queries in parallel, demonstrates a significant reduction in overall processing time. Even with large amounts of text, this method can reduce the total generation time by almost three times compared to the basic approach, which indicates its high efficiency and practical value.

# Conclusion

The experiments confirmed the effectiveness of the proposed method of audio generation using the dynamic programming algorithm. Compared to the basic method, where audio is generated by sequential requests of 4096 characters, the optimized approach allowed us to reduce the total processing time by almost three times.

Thus, the use of dynamic programming significantly improves system performance and reduces delays when users access audio materials.

##### References

1. Pinedo. M. Scheduling Theory, Algorithms, and Systems. New York: Springer, 2012. 69-109 p.
2. Roughgarden T. Roughgarden Algorithms Illuminated (Part 3): Greedy Algorithms and Dynamic Programming. New York: Soundlikeyourself Publishing, 2019. 1-21 p.
3. K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms. Chichester, UK: Wiley, 2001.
4. M. Dorigo and T. Stützle, Ant Colony Optimization. Cambridge, MA: MIT Press, 2004.
5. J. Kennedy and R.C. Eberhart, Swarm Intelligence. San Diego: Morgan Kaufmann, 2001.
6. D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.