โครงงานเลขที่ วศ.คพ. S020-1/2565

เรื่อง

วิชิวิกสำหรับตรวจการใช้ไวยากรณ์ภาษาไทย

โดย

นายอภิสิทธิ์ ฤทธิ์เรืองโรจน์ รหัส 620610820

รายงานนี้เป็นส่วนหนึ่งของวิชาสำรวจเพื่อโครงงาน
ตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
ปีการศึกษา 2565

PROJECT No. CPE S020-1/2565

WYSIWYG typing assistant and Thai grammar checker

Apisit Ritreungroj 620610820

A Report Submitted in Partial Fulfillment of Project Survey Course
as Required by the Degree of Bachelor of Engineering
Department of Computer Engineering
Faculty of Engineering
Chiang Mai University
2022

หัวข้อโครงงาน	: วิชิวิกสำหรับตรวจการใช้ไวยากรณ์ภาษาไทย
	: WYSIWYG typing assistant and Thai grammar checker
โดย	: นายอภิสิทธิ์ ฤทธิ์เรืองโรจน์ รหัส 620610820
ภาควิชา	: วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	: อ.ดร. ชินวัตร อิศราดิสัยกุล
ปริญญา	: วิศวกรรมศาสตรบัณฑิต
สาขา	: วิศวกรรมคอมพิวเตอร์
ปีการศึกษา	: 2565

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ ได้อนุมัติให้โครงงานนี้เป็นส่วน-หนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต (สาขาวิศวกรรมคอมพิวเตอร์)

..... หัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์

(รศ.ดร. สันติ พิทักษ์กิจนุกูร)

คณะกรรมการสอบโครงงาน

..... ประธานกรรมการ

(อ.ดร. ชินวัตร อิศราดิสัยกุล)

..... กรรมการ

(ผศ.ดร. กานต์ ปทานุคม)

..... กรรมการ

(อ.ดร. พฤษภ์ บุญมา)

Contents

Chapter 1
Introduction

1.1 Project rationale

Nowadays, there are many advancements in computational linguistics. in English, there are so many grammar checker vendors out there, e.g., Grammarly [9]. But there are some differences between Thai and English as word segmentation remains a fundamental challenge in the Thai language since it does not require spaces to separate words. And the process of segmentation can cause ambiguity if the context provided is not enough, such as 'กอดอก' is be segmented into either 'กอ|ดอก' or 'กอด|อก'. This is, so far, challenging to create a grammar checker for Thai.

This project dedicates to constructing a basic grammar checker for the Thai language. It is planned for normal people to freely use and access this service publicly. In this report, all the concepts and implementation details are explained. The core language model is implemented in highly optimized algorithms, and partially integrated with PyThaiNLP modules [15]. The report also include details about the process of collecting dictionary data from the Royal Institute Dictionary [30] through the process of serving the suggestions to users.

1.2 Objectives

- To create a Thai grammar checker service.
- To create a service that can handle at least 1,000 requests per second concurrently and respond with correction suggestions spontaneously.

1.3 Project scope

The project is a web-based application capable of performing the following tasks:

- autocorrection [25]
- grammatical error detection
- character arrangement, e.g., the upper vowel and the lower vowel cannot sit at the same consonant or position
- wrong or outdated character usage, e.g., usage of 'เ เ' (double 'เ') to act as 'แ' is incorrect
- sentence, word, and conjunction spacing
- punctuation usage.

There are also tasks we do not consider to be in the scope, as the project primarily focuses on the foundation of the language grammar. This does not mean that they will not be implemented in further development. These tasks include

- English compatibility
- paraphrasing [29] and clarifying the sentence
- reference ambiguity and unclear detection
- monotonous sentences detection
- positive or negative sentences detection
- sarcastic sentences detection
- overall tone of the sentences detection
- named-entity recognition and wikipedia linking
- plagiarism detection in writings
- copywriting [27]
- parsing unstructured text
- summarization [26]

The result will be sent as suggestions to the end-user, who will select the most appropriate one.

1.3.1 Hardware scope

All PCs, tablets, and smartphones that can use a modern browser and access the internet. A modern browser is a browser which at least compatible with HTTP2 connections and TLS1.2–1.3 encryption, and generally supports modern JavaScript syntax (ES6) and modern media compression, e.g., `.webp` (image) and `.woff2` (font).

1.3.2 Software scope

The project is a web application, available as a Chrome-based browser extension.

1.4 Expected outcomes

- Normal people are able to access the service freely via the internet.
- Reducing proofreading time people spend on writing.
- Improving overall writing quality.

1.5 Technology and tools

A variety of tools are required in this project for building complex application services. Since the services in this project are cloud-based, the technology mainly involves web service and practices.

1.5.1 Hardware technology

A single DigitalOcean's cloud-based IaaS instances (Droplets) for back-end service deployment. The instance uses shared Intel 1vCPU, 1GiB / 25GiB SSD disk.

1.5.2 Software technology

Programming languages

- Rust [17]
- TypeScript [23]

Tools

- Docker (container and deployment) [6]
- Debian (Linux distro for deployment) [4]
- DigitalOcean (cloud platform) [5]
- Solid (front-end framework) [20]
- SCSS (style sheet language, compiled to CSS) [7]
- Vite (front-end build tool) [24]
- Actix (back-end framework) [1]
- SeaORM (back-end ORM) [19]
- PostgreSQL (RDBMS) [14]
- DBML (DSL for define database schemas) [3]
- Protocol Buffers (payload serialization language) [8]

Network and architecture

- HTTP2 [28] with TLS1.3 [10]
- token-based authentication [12]
- client-server architecture [22]
- automated deployment pipeline [13]

1.6 Project plan

Task	Apr 2022	May 2022	Jun 2022	Jul 2022	Aug 2022	Sep 2022	Oct 2022	Nov 2022	Dec 2022	Jan 2023	Feb 2023
Planning and research											
Dictionary and language module implementation											
Back-end implementation											
Front-end implementation											
Testing and evaluation											
Revision from feedback											

1.7 Roles and responsibilities

Apisit Ritreungroj is responsible for every aspect of the project, i.e., planning, research, design, management, and execution.

1.8 Impacts of this project on society, health, safety, legal, and cultural issues

This project aims to be a real-world solution for facilitating Thai proofreading in various types of writing, to reduce time spent on proofreading each publication and writing. There is a concern for people who dedicate themselves to this career that the solution might ruin their entire job [16].

Chapter 2
Background Knowledge and Theory

Our grammar checking service needs to have modules that can process the Thai language, e.g., tokenization, named-entity recognition (NER), and dependency parsing. Those modules are contributed to PyThaiNLP [15], an open-source NLP toolkit for Thai. The PyThaiNLP module is well-equipped with many tools, covering all the needs of the project. The drawback is that the module is implemented in Python, which is resource-consuming and slow [2] and not suitable for low-end servers for cost-saving purposes. To avoid this drawback, the project will use a compiled programming language, Rust [17]. Therefore, the project reimplements some parts of the PyThaiNLP module, e.g., the TCC tokenization engine, into the Rust implementation as it has solid grammar rules to work with and also gain a performance boost.

Most of the language validation modules in the project are created from scratch with Rust. The part of tokenization, whether the grapheme tokenization or TCC tokenization process, each can break into grammar rules, and the engine needs to validate each of them in smaller parts, which PyThaiNLP does not have this validation process. The project aims to integrate some of the PyThaiNLP modules, such as the dependency parser, as it requires a lot of time to reimplement a precise one and test for its correctness. Regarding the performance of Python or Rust, the final goal is to create a not-too-slow Thai grammar checker with an acceptable usability level.

2.1 Fundamental Thai grammar knowledge

The fundamental Thai grammar knowledge uses in this project, e.g., the study of word spelling, word usage, tonal marks usage on a consonant, and punctuation usage. In order to correct the misspelt words in writing, the study of correcting words is applied. A misspelt word such as ‘กระเพรา’ needs to be corrected as ‘กะเพรา’. In a formal context, punctuation usage is also very important, for example, the repetition character ‘ๆ’ needs to have a single space before and after the character. Furthermore, the formation of sentences is studied in the project as words need to form together into sentences.

2.2 Natural language processing (NLP)

The practices of NLP are involved generally in the project. The process of text preprocessing is applied in the part of fetching words from the Royal Institute Dictionary. The fetched words are embedded in various HTML formats. So those words need to be processed and stored in the structured form before using them. Word segmentation is also applied in selection of techniques used in this project, e.g., maximal-matching algorithm.

2.3 Compiler design

Compiler design knowledge is applied commonly in this project. It provides a guideline for the design process to be efficient and smooth. Parts of the front end of compiler design, e.g., tokenization, usage of the deterministic finite automaton (DFA), and abstract syntax tree (AST) construction, are primarily used in the project.

2.4 Cloud engineering and web development

Web development is used to create an interactive interface for users to interact with the service. The project used a lot of web development practices, e.g. framework usage, built tools usage, styling and layouting, accessibility, and responsive design.

The cloud knowledge is involved in the process of the web client and back-end service deployment, and also in security configuration to the containers and the virtual machines.

2.5 UX/UI Design

The knowledge applies to the practices of designing an easy-to-use and appealing to attract users. The design knowledge mainly applied in this project includes color theory, usability heuristics, and UX design psychology.

2.6 CPE knowledge used, applied, or integrated in this project

- 261207 – CPE Lab: applied in web service development
- 261217 – Data Structures: applied in creating performant data flow and dictionary datastore design
- 261218 – Algorithms: applied in backend process optimzation
- 261342 – Database: applied in database schemas design
- 261361 – Software Engineering: applied in practices during development process
- 261335 – Networks: applied in designing the secure service over the internet
- 269462 – Human–Computer Interaction: applied in usablity heuristics of the application design

2.7 Extracurricular knowledge used, applied, or integrated in this project

The obvious one that cannot lack is *linguistics* knowledge. Huge linguistics, especially Thai linguistics, was applied in this project to create programming that can analyze a whole sentence and even a whole article. The second is *data science* knowledge to deal with huge amounts of data, e.g., linguistics corpus and dictionary manipulation of over 80,000 entries.

The last is the knowledge of *natural language processing* (NLP), used during the development process of the project, e.g., text preprocessing after fetching words from the Royal Institute Dictionary. Other minor knowledge applied in the project includes *UI Design*; knowledge about the programming language used in the project, e.g., *Rust*; and the practice of *DevOps*.

Chapter 3
Project Structure

There are many components and modules that comprise the entire grammar checking service. Each is a standalone service communicating with one another to form the a seamless service.

The service consists of 4 main modules: the dictionary management system, the language core, the back-end service, and the front-end service. The dictionary management system is responsible for managing the corpus for the language core module. The language core is where all the language validation processes. The back-end is a part where delivers the language core to the internet. Finally, the front-end is where the user interacts with the service.

3.1 Dictionary management system

This module enables other services to communicate with the dictionary. It performs querying words and adding new words on the request. There is only a single dictionary management system per back-end service. The system performs word querying and word modifying in the database, which the back-end service cannot do directly.

3.1.1 Dictionary file

The dictionary file is a file format for the purpose of debugging before being sent to store in the database. The structure of the file is `.yaml` [32] as it is easy to read, and also enables the maintainer to add or modify the file easily. Note that the file is manually edited. The initial structure shows below.

```
<word>:
  - pron?: <string>;
    role?: <string>;
    form: <verb | adverb | pronoun | noun | conjunction | preposition | interjection
    meaning: <string>;
    origin: <string>;
    syns?: <string[]>
  - ...
```

1. <word>: is a word. A word can have many definitions.
2. pron: is the pronunciation of a word, e.g., the word ‘ทฤษฎี’ pronounces as ‘ทริด-สะ-ดี’.
3. role: is the role of a word in the specific field, e.g., the word ศาลฎีกา is in role of law.
4. form: is a part of speech of a word.
5. meaning: is the definition explanation of a word.

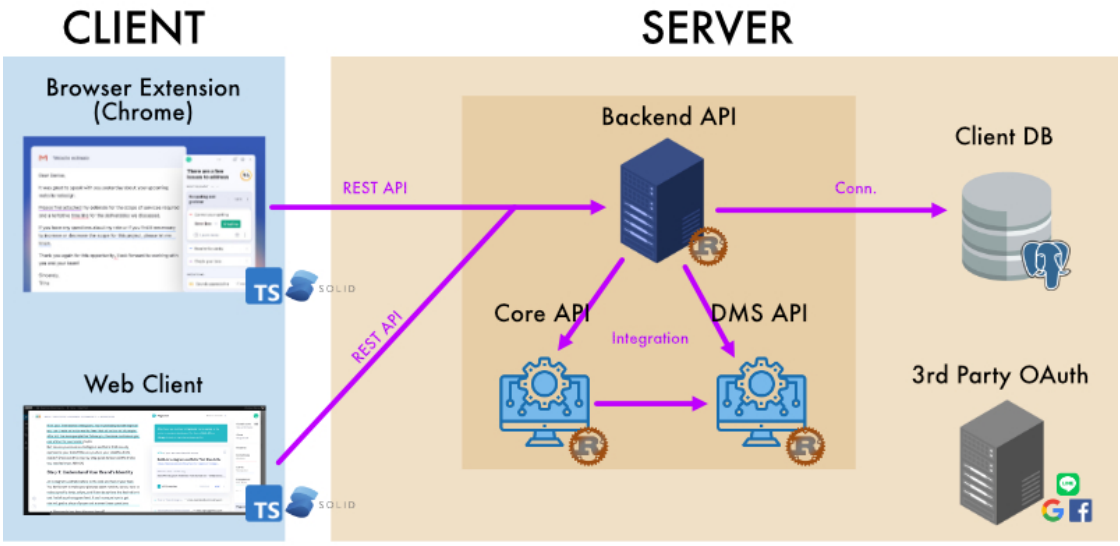

Figure 3.1: The overview architecture of the service

6. origin: is the origin of a word, e.g., the word 'คอมพิวเตอร์' originates from English.
7. syns: is an array of synonym words related to its definition.

3.2 Language core

The language core is the core service, which validates users' text and returns the result as correction suggestions. To achieve real-time results and high concurrency in low-end servers, it is implemented in the Rust programming language as it is performant [33], memory-safe [18], and low in memory footprint [21]. The process is defined in the from of pipeline flow split into stages.

3.2.1 Validation pipeline

The validation pipeline analyzes text in stages to validate its correctness. There are three stages in the pipeline: tokenization, parsing, and contextual analysis.

Tokenization

The first stage is tokenization, where every single word is tokenized. Tokenization itself is split further into three steps: Grapheme tokenization, TCC tokenization, and word tokenization.

1. Grapheme tokenization is the first step of the validation process. It splits text into *graphemes*, i.e., vertical tokens; for example, the word 'น้ำท่า' will be split into น้ำ | ท่ | า.

This step validates arrangements of vertical vowels and consonant characters; for example, ‘จิ’ would be incorrect, since the upper vowel and the lower vowel cannot appear simultaneously in a grapheme.

2. TCC (*Thai character cluster*) tokenization follows. It joins grapheme tokens into TCC tokens. TCC is the combination of the horizontally independent vowels, such as ‘า’ and ‘แ’, and one or more consonant characters together. These vowels cannot appear alone and must connect to a consonant character. Examples of a TCC token: ‘เพลิง’ is เพ|ลิ|ง, ‘น้ำท่า’ is น้ำ|ท่า. This step also validates the character arrangements in the horizontal axis; for example, ‘โเ’ or ‘เา’ would be definitely wrong grammatically.
3. Word tokenization is the last step of tokenization. It forms TCCs into words by using maximal-matching algorithm against the words in the dictionary. This step involves autocorrections, which attempt to correct words if they do not match any word in the dictionary. Word tokenization can validate extra characters that could break the whole sentence, e.g., ‘เพลิงงไหม้’, the extra character ‘ง’ results in interruption of the pipeline.

Parsing

This stage tries to form word tokens into a grammatical structure, also known as an *abstract syntax tree* (AST). There will be predefined rules of Thai language grammar. If there is a token that does not fit into the grammar rule, The pipeline will throw error suggestions, stop the whole process, or skip to the next word for more contextual information. For more complex analysis, this stage might involve the usage of machine learning because, so far, without machine learning, it cannot recognize any name or entity in the sentence. This could result in faulty corrections. In this stage, sentences are also formed.

Contextual analysis

This is the final stage of the pipeline. It takes the so-called “ASTs of sentences” and analyzes them one by one. The stage is responsible for defining the types of a sentence, e.g., simple, compound, complex-compound, only performing the basic level of analysis. No validation process involves at this stage.

After the final stage is completed, suggestions and errors collected during those three stages in the pipeline will be sent as output. Note that one of the stages could interrupt the whole process early and send results immediately. Some errors need to be corrected before proceeding to the next pipeline stage.

3.3 Web back-end

The back-end service delivers the language core to the open internet. It communicates with clients (the front-end services), receiving the users' input, and returns and compresses the result from the language service. It is implemented using Actix as the web framework of the Rust programming language for multithreading and concurrency. It also enables HTTP2 and TLS1.3 for secure connection over the internet. The request-response payload is encoded in a highly compact binary for reducing payload size, thanks to Protocol Buffers. This part also accounts for the business logic; most of the users' data are processed here.

3.4 Web front-end

The user interface at the front end is a web-based service with which users interact with to communicate with the back-end service. It is implemented in Solid [20] as a framework of choice and TypeScript as a programming language. The Solid framework is used as it improves the speed of the front-end compared to the popular framework like React [11]. The front-end service is responsible for chunking sentences, preprocessing, indexing, caching the back-end responses, and displaying the easy-to-interact-with results from the back-end.

3.4.1 Browser extension

The browser extension facilitates users to validate their messages and articles at any site where the WYSIWYG editor exists, e.g., Facebook or Gmail. Note that, as the project has a limited time, the extension is designed to be available for Chrome only since it is the most used browser [31]. Other browsers, i.e., Safari, Firefox, and Edge, will be developed in further development for a variety of accessibility.

Chapter 4
System Evaluation

The evaluation process consists of two parts: the validation testing and performance test for the language validation, and the usability test for users.

4.1 Evaluating language validation

To evaluate the language validation service, validation and performance tests will be performed. Both tests are not compared with any other module, e.g., PyThaiNLP.

4.1.1 Validation test

The validation test evaluates against predetermined sets of text containing faults. The test suite contains at least 1,000 entries of texts, phrases, and articles, to evaluate the correctness and validity of the service.

4.1.2 Performance test

The performance test is a stress test on the capability of the entire service, including the core module, the back-end service, and the front-end service.

The language validation module

This part performs throughput-output testing of the core module to obtain an insight into the following metrics:

- text length capacity
- processing time on text inputs of various length
- the relationship between processing time and text length

The back-end service

This part performs request-response throughput-output testing, including

- request-per-second testing.
- database query testing
- dictionary query testing

The front-end service

This part tests the WYSIWYG editor; to determine the indexing and caching performance related to text length that they impact on the site.

4.2 Usability test

A group of voluntary users in CPE of at least ten will perform the usability test. There will be a set of predefined tasks for them to do. The service will gain a usability score if the users can perform each task without any problem.The users can comment freely and openly on their thoughts about the service. Their feedback and comments will be kept for improving the project within the project's timeline.

References

[1] Actix developers. What is an orm, 2022. [Online; accessed 12-Octuber-2022].

[2] Nadav Altshuler. Why is python so slow? - tackling python's performance issues, 2019. [Online; accessed 12-Octuber-2022].

[3] DBML developers. Dbml - database markup language, 2022. [Online; accessed 12-Octuber-2022].

[4] Debian developers. Debian – about debian, 2022. [Online; accessed 12-Octuber-2022].

[5] DigitalOcean developers. Digitalocean – the developer cloud, 2022. [Online; accessed 12-Octuber-2022].

[6] Docker developers. Docker overview | docker documentation, 2022. [Online; accessed 12-Octuber-2022].

[7] GeeksforGeeks authors. What is the difference between css and scss?, 2022. [Online; accessed 12-Octuber-2022].

[8] Google developers. Overview | protocol buffers | google developers', 2022. [Online; accessed 12-Octuber-2022].

[9] Grammarly developers. Free gramamr checker by grammarly, 2022. [Online; accessed 12-Octuber-2022].

[10] Brian Jackson. An overview of tls 1.3 – faster and more secure, 2022. [Online; accessed 12-Octuber-2022].

[11] Iniubong Obonguko. Solidjs vs. react: Comparing declarative ui libraries, 2022. [Online; accessed 14-Octuber-2022].

[12] okta authors. What is token-based authentication?, 2022. [Online; accessed 12-Octuber-2022].

[13] PagerDuty authors. What is a deployment pipeline?, 2022. [Online; accessed 12-Octuber-2022].

[14] PostgreSQL developers. Postgresql: The world's most advanced open source relational database, 2022. [Online; accessed 12-Octuber-2022].

[15] PyThaiNLP authors. Pythainlp, 2022. [Online; accessed 12-Octuber-2022].

[16] replacedbyrobot authors. Will “proofreaders and copy markers” be replaced by robots?, 2022. [Online; accessed 12-Octuber-2022].

[17] Rust developers. Rust programming language, 2022. [Online; accessed 12-Octuber-2022].

[18] Deepu K Sasidharan. Why safe programming matters and why a language like rust matters, 2022. [Online; accessed 12-Octuber-2022].

[19] SeaORM developers. Seaorm an async & dynamic orm for rust, 2022. [Online; accessed 12-Octuber-2022].

[20] Solid developers. Solidjs · reactive javascript library, 2022. [Online; accessed 12-Octuber-2022].

[21] Michael Spencer. What are the greenest programing languages?, 2022. [Online; accessed 12-Octuber-2022].

[22] The Editors of Encyclopaedia Britannica. client-server architecture, 2022. [Online; accessed 12-Octuber-2022].

[23] TypeScript developers. Typescript: Javascript with syntax for types., 2022. [Online; accessed 12-Octuber-2022].

[24] Vite developers. Overview | vite, 2022. [Online; accessed 12-Octuber-2022].

[25] Wikipedia contributors. Autocorrection - wikipedia, 2022. [Online; accessed 12-Octuber-2022].

[26] Wikipedia contributors. Automatic summarization - wikipedia, 2022. [Online; accessed 12-Octuber-2022].

[27] Wikipedia contributors. Copywriting - wikipedia, 2022. [Online; accessed 12-Octuber-2022].

[28] Wikipedia contributors. Http/2 - wikipedia, 2022. [Online; accessed 12-Octuber-2022].

[29] Wikipedia contributors. Paraphrase - wikipedia, 2022. [Online; accessed 12-Octuber-2022].

[30] Wikipedia contributors. Royal institute dictionary - wikipedia, 2022. [Online; accessed 14-Octuber-2022].

[31] Wikipedia contributors. Usage share of web browsers - wikipedia, 2022. [Online; accessed 14-Octuber-2022].

[32] Wikipedia contributors. Yaml - wikipedia, 2022. [Online; accessed 14-Octuber-2022].

[33] Serdar Yegulalp. What is rust? safe, fast, and easy software development, 2021. [Online; accessed 12-Octuber-2022].