Everyone can NLP A User-defined Well Visualized NLP Training Platform

by Yuan Ren

Submission date: 27-Jul-2022 11:49PM (UTC+0800)

Submission ID: 1875849885

File name: 231658_Yuan_Ren_Everyone_can_NLP_A_User-defined_Well_Visualized_NLP_Training_Platform_2587533_1415113265.pdf (3.47M)

Word count: 10290

Character count: 55543

The University of Hong Kong

Faculty of Engineering

Department of Computer Science

COMP7705

Project Report

Everyone can NLP: A User-defined Well Visualized NLP Training Platform

Submitted in partial fulfillment of the requirements for the admission to the degree of Master of Science in Computer Science

By

CAI Mingzhu(3035906136)
FAN Yixuan(3035918842)
REN Yuan(3035905429)
XU Xuyang(3035905077)

Supervisor: Dr.KONG Lingpeng

Date of submission: 30/07/2022

11

Abstract

In recent years, Natural Language Processing (NLP) has been widely leveraged in many fields, but the difficulty of processing texts and grasping NLP techniques has become a barrier to implementation.

The project focuses on people with no formal training experience in artificial intelligence (AI) coding and AI development but who need to build NLP models to solve text processing problems. The project aims to develop a well-visualized and easy-to-operate one-site platform to support users in developing custom NLP models.

The platform proposes to cover the basic need of NLP workflow that contains data processing, model training, result evaluation, and model deployment. The platform comprises three main sections: data management, model architecture design, and model training. The functional modules help users construct their NLP models from a more technical and detailed view and understand the data and the result better.

This project highlights the addition of fine-tuning within the model to the regular construction of the NLP model. As a result, users can view each layer of the model and make custom modifications, thus increasing the freedom of the model and reducing the difficulty of model construction.

Declaration

I hereby declare the originality of this dissertation thesis. The contributions of others involved are indicated clearly within the text of my work.

Acknowledgements

Our sincere and hearty thanks and appreciations go firstly to our supervisor, Dr.Kong Lingpeng, whose suggestions and encouragement have given us much insight into the project. Studying under his guidance and supervision has been a great privilege and joy. Furthermore, it is our honor to benefit from his personality and diligence, which I will treasure our whole life. The gratitude to him knows no bounds.

Contents

1 Introduction

1.1 Background

Natural Language Processing (NLP) is a branch of Artificial Intelligence and Linguistics whose primary research goal is to explore how to process and use natural language. NLP is designed to solve tasks such as cognition, understanding, and natural language generation.

The NLP modeling platform allows users to build their NLP models. Nowadays, NLP models are needed in many scenarios, but most users do not have the experience or knowledge of Artificial Intelligence, so many companies provide corresponding NLP model platforms, such as Baidu, Alibaba.

The built models can be used in various fields, such as text summarization, lexical annotation, and central word extraction. However, the current NLP modeling platforms focus on making it easier for users to use the models provided by the platforms rather than the models trained by users with less experience. Users still face the problem of not being able to train models independently.

Therefore, this project aims to solve this problem by implementing a well-visualized model-building platform to help users train their models with no code. The main difficulties in the project are how to break down the algorithm into easy-to-understand functional blocks and how to make it more user-friendly on the website.

1.2 Achievements

In the project, our team developed an NLP model building platform for users with less experience to build customized NLP models and use them for classification tasks.

The platform achieves more user-friendly and more useful in the following aspects.

1. High degree of freedom.

The platform mainly solves the problem of building a high degree of freedom model and assists users in the design of different layers in the model while the users can select the specific complex NLP model.

2. Well-visualized.

The platform offers various kinds of visualization in all aspects. The user can view the graphical results of the uploaded dataset, interact with the model canvas and receive the training model details through the TensorBoard.

3. Easy to use.

Since the platform is well-visualized, the user can accomplish the model building through the GUI interaction without coding.

4. One-stop service.

The platform implements the functions of data upload, model training, and model prediction so that users can complete the whole process of standard model building, from uploading data to obtaining model prediction results on the platform.

5. SOTA NLP Algorithm.

The platform introduces the large-scale pre-trained language model with advanced modifications like Bert. The models constructed by the platform can achieve high accuracy through the evaluation experiments.

6. Compatible with multiple language data.

The platform can handle different types of language data, from data visualization to model prediction.

1.3 Overview of Report

The report includes seven parts to explain the platform entirely.

The Introduction introduces the background showing the current situation and the main problem faced by similar machine learning platforms and provides a basic view of the achievements of our platform's functions.

The Literature Review mainly discusses the common deep-learning-model-building platforms in the markets and their weaknesses. The significance of NLP and the earlier machine-learning-model-building platforms are also presented.

The System Design and the System Analysis represent the methodology included in the project.

The System Design consists of the procedure of how the platform works, the design of the functional module, and the technology selected to implement the platform.

The System Analysis focuses more on the different ends of the system and how they cooperate in implementing the system integrity.

The Discussion introduces the methodology to evaluate our system, especially our provided model modifications. This part will show the performance of the modified model.

The Conclusion consists of the achievements and problems the team still encounters.

The Future Work part will introduce the long-term schedule and how the current problems can be solved in the future.

2 Literature Review

NLP is a hot research topic that has drawn the attention of the Artificial Intelligence (AI) community for the past few decades [1]. However, for the group of people with little experience in coding but who need to handle large amounts of text data, a platform that supports users to develop custom AI models will be helpful.

The earliest concept that novices can utilize ML to process text tasks is proposed by Automated machine learning (AutoML) in 2013[2]. AutoML aims to create software that helps people with little coding background to conduct the parts of the machine learning processes, including preprocessed text data, choosing appropriate models and optimizing hyperparameters, etc. Nevertheless, users have problems getting satisfying models only using traditional ML methods. Low robustness and efficiency models constructed by traditional ML methods perform unstably corresponding to different application scenes [3]. Furthermore, models built by Deep Learning (DL) methods can perfectly solve the problem caused by applying traditional ML models.

In recent years, with the development of Deep Learning (DL), many technology companies such as Google and Alibaba have developed open-source NLP platforms to provide more robust and efficient models based on DL methods. For example, the Language Studio platform designed by Alibaba in 2019 provides pre-trained language models based on Transformer, like Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT) to support users dealing with complex text tasks [4]. Transformer applies a self-attention mechanism to focus on the most useful words in predicting words in a sentence. With the self-attention mechanism, Transformer processes an input sequence of

words all at once and directly maps relevant dependencies between any two words in the context regardless of distances. Therefore, Transformer is highly parallelizable and efficient, which can also train much larger models at a faster rate and use contextual clues to solve ambiguity issues that plague text [5]. Although platforms like Language Studio have provided efficient models, it is worth noting that current platforms are not quite user-friendly as they do not support users to visualize their model architectures and only use text instructions to guide users to tune models. Generally speaking, platforms' instructions contain many professional terminologies, meaning users with little computer science background may have difficulty understanding how to modify model architectures or tune hyper-parameters. In addition, these platforms do not store users' datasets, meaning users should repeatedly upload datasets even though the same can be applied to different tasks.

Current NLP platforms generally have problems providing inexperienced users with efficient and easy-to-operate models, as they fail to support visualization of model architecture and storage of users' datasets. To solve these problems, our project aims to utilize pre-trained language models, including BERT and GPT, as basic models to provide users with customized AI algorithms capabilities. Meanwhile, we also introduce visualization methods to support users in visualizing model architecture and modifying models online. Besides, we allow each user to own their own training space to store their data information.

3 System Design

3.1 System Overview

3.1.1 Software Design Principle

According to the system business logic requirements and the basic guidelines of software development, the design of this system mainly follows the following SOLID principles.

1. Single Responsibility Principle:

The system is separated into three main sections, including multiple functional modules. Each functional module can implement a specific single function so the system's complexity can be lower.

2. Open/Closed Principle:

To make the system easier to refine, after finishing the framework, the modifications of the system follow the OCP to refine the current functions and add new modules.

3. Liskov Substitution Principle

Following ISP, the inheritance through the codes has been protected to make the project easier to modify and reuse.

4. Interface Segregation Principle

To lower the coupling of the system, ISP should be followed to limit the interface as smaller and more specific as possible.

5. Dependency Inversion Principle

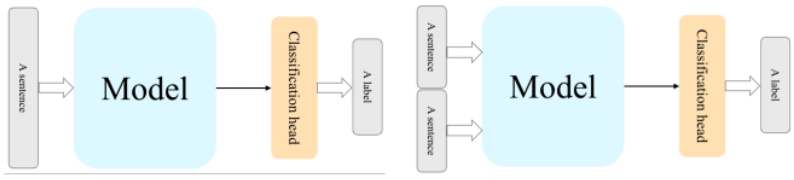
The system will become more stable and modifiable to ensure that the different modules depending on the abstract object.

3.1.2 Algorithm Design

Our platform is designed for users to complete single sentence classification and dual sentence classification, as shown in Figure 1.

According to open source datasets registering in Hugging Face[1], text classification is the most common task in natural language processing, having 443 datasets. In contrast, the second most popular task, question answering, has 167 datasets. In addition, many kinds of architectures have classification ability that can be used in our platforms, such as TextCNN [6].

The standard sentence classification, including spam emails detection, sentiment analysis, and hate speech detection, takes one sentence as input and returns one label through models. With the increasing complexity of models, more intricate classification problems can be solved, such as semantic similarity classification and relation detection, which takes two sentences as input and returns one label. Therefore, our platform aims to support single and multiple inputs to maximize the boundary of our ability.



(a) Single sentence classification (b) Dual sentence classification

Figure 1: Model Input

We divide the process of a task into training, evaluation, and prediction. A serviceable model needs to be trained to fit the dataset. The evaluation can test the

[1]https://huggingface.co/datasets

generalization ability of the model to an unseen dataset. After adequate training and evaluation, models can be deployed to forecast data.

We provide informative results when the task completes. For training tasks, we allow users to monitor the training state by TensorBoard, such as learning rate, loss, and evaluation metrics. TensorBoard[2] is a tool for providing measurements and visualizations during training. However, using TensorBoard is limited and challenging for users without coding experience as it requires a suitable Python environment. In addition, we can only utilize TensorBoard locally or register an account on Tensorboard website to upload data for visualizations. Our platform completes all the processes needed to use TensorBoard by running TensorBoard on our server and providing a visiting link for users. For evaluation tasks, we offer loss, accuracy, and confusion metrics in an attempt to give a comprehensive understanding of model performances. For prediction tasks, users can download the prediction results of their trained models.

Our platform adopts the most favorite model in NLP – BERT [7] to conduct classification tasks compared to existing machine learning websites like AutoML using traditional machine learning algorithms. The details of our model will be stated in section 4.3.

[2]https://www.tensorflow.org/tensorboard

3.1.3 System Procedure

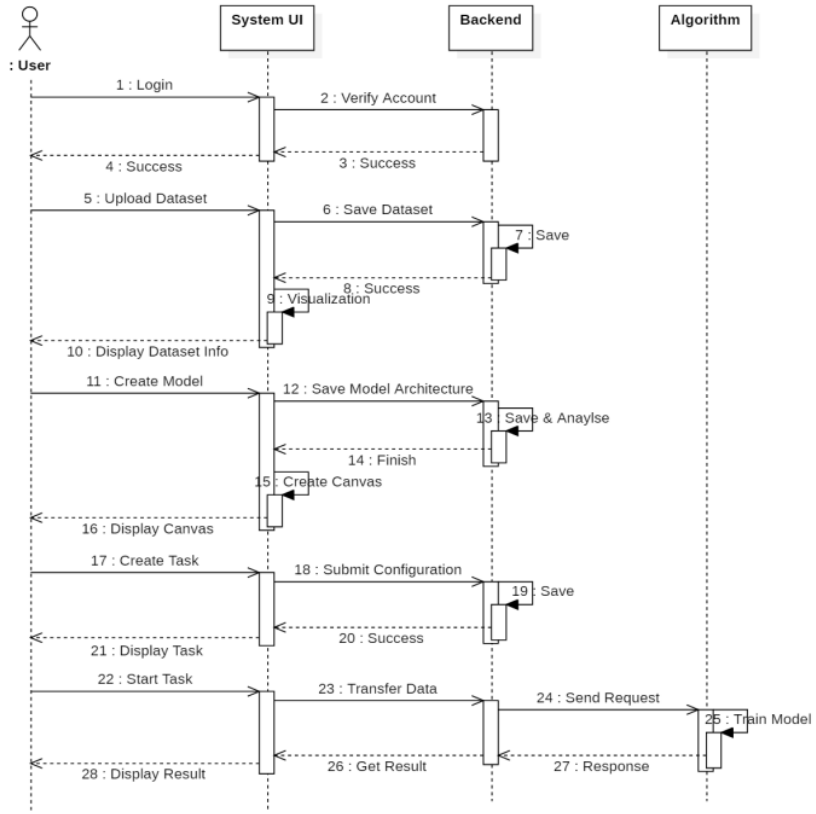


Figure 2: System Procedure

As a platform aimed to help users build the NLP models, the procedure is explained to show how the platform works to build the model.

The user should log in to the website to start the process. If the user has never had an account, the system also provides the section to register a new account. Then the website sends the information to the back-end to verify the account. If the authentication passes, the sign of success will return from the back-end through the website to the user.

An entire process can be divided into uploading datasets, designing models, and

starting tasks:

1. It is needed to upload a specific data file in the CSV format supplied by the user.

2. The website sends the dataset to the back-end for saving and detailed processing once the dataset is configured well and confirmed.

3. The saved dataset can be viewed on the website, and the user can also check the specific charts to broaden the view of the data.

After preparing datasets, the user should define the details of the customized model, including the model's name, model type, and model pre-trained parameter. The model architecture and basic model information are saved to the server as the model configuration file. The default model architecture defined by the user is illustrated on the canvas. The user can graphically modify the architecture to meet the need. Furthermore, the user can examine the visualized model layers on the canvas and change the defined model at any time.

After the data and the model configuration are finished, the user can create the various tasks. The tasks include the training task, the predicting task, and the evaluating task. The task section is designed to use the model. The specific task configuration will be saved in the server for further operations, which can be checked on the website after creating.

The final step is to start the task. The selected data and model configuration will be sent to the algorithm-end. For different tasks, the result will be different. For example, the prediction tasks' the heatmap of the confusion matrix, and the prediction result table will be shown on the result page. The user can also download the result CSV and the log files for further analysis.

3.2 Functional Module Design

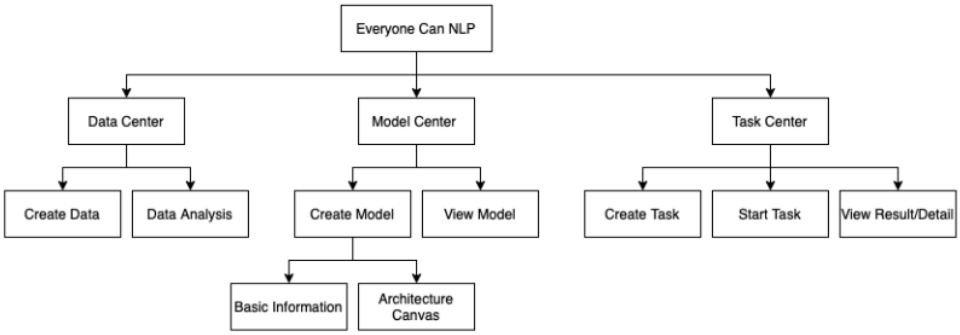


Figure 3: Functional Modules

According to the system procedure, the system design is separated into 3 main functional modules: data center, model center, and task center.

1. Data Center

Data Center consists of two subsections - create Data and Data Analysis.

(1) Create Data is used to upload the user's dataset processed in CSV format and register the dataset in our system.

Users are required to define the dataset's name and select a local file from the user's computer. The dataset is planned to be used in single-sentence or dual-sentence classification. Therefore, except for the basic dataset information, such as the name, the type of the dataset should be decided. The type of the dataset represents the type of task that the data can be used. The single sentence type means the task has 1 input and 1 output used in single sentence classification. The dual sentence type means the task has 2 inputs and 1 output used in dual sentence classification. The file of the dataset usually contains redundancy data that will not be used during the model training. Users are expected to determine the name of columns to be used as input and output.

(2) Data Analysis includes three charts – word cloud, pie chart, violin chart, to help the users have a deeper view of the data.

- Word cloud: Word cloud is a graph showing the combination of words in the dataset. The size of a word represents the frequency of words, while the color of a word is determined by the source of the word, i.e., which class the word comes from. Word cloud displays the main topic about the content, general features of data according to the labels, and the overview of the data. Word frequency indicates the main topic of content. Combined with the source of words, users can grasp an overview of the data and discover the general feature of the data class.

- Pie chart: A pie chart presents the distribution of labels. Each partition in a pie chart shows the number of data from the same class. Users can digest the distribution features of labels and have an intuitive view of whether the dataset is balanced.

- Violin chart: The violin chart in our platform demonstrates the distribution of the content length counted by the number of words. The violin chart exhibits the sentence length's mean, minimum, maximum, and variance. The distribution of the content length can help to pre-process the data for the model building since the BERT cannot handle the sentence with infinite length and requires a maximum input length. With the information of the longest length of the content, [the maximum length of the input sentence] can be decided in the earlier stage.

2. Model Center

Model Center consists of Create Model and View Model.

(1) Create Model offers a brand-new way to generate the model configuration.

There are two parts in the model configuration. One part is the basic information of the model, such as a model name, model description, the basic model, and the pre-trained parameter. The other is to design the model architecture through the canvas. The canvas offers GUI for users to design the details of the model leading to the high freedom.

(2) View Model is used to have an overview of the users' created models.

The model overview displays a simple schematic diagram indicating the number of layers in the model according to the per-trained parameter selected by users. Besides the model overview, users can dive into the model to find the details of the model architecture and alter the architecture based on their needs.

3. Task Center

Task Center aims to use the data and the model created in the previous steps to run the model and obtain the results. In Task Center, three types of tasks are introduced.

Training Task receives a dataset as training data from the data center and a model from the model center and fine-tunes the model to fit the dataset.

Evaluation Task receives a dataset from the data center and a model fine-tuned by the training task. The performance of a fine-tuned model is evaluated from several aspects, such as the loss, accuracy, and confusion matrix. The confusion matrix shows the distribution of the bad cases, which is valuable for detecting the relation between bad cases and locating the reason for model mistakes.

Prediction Task utilizes the model already trained in the previous task to predict the unlabeled data.

(1) Create Task

The task should be created by selecting the preliminary data, and model. There is no need to specify the class number as our platform conducts it automatically for users. Except for the basic information and the selection, users need to determine the fine-tuned model parameters for evaluation and prediction tasks.

(2) Start Task is used to start the user's task. The platform also provides the function to stop the running task as soon as the user wants.

(3) View Results/Details

Since the platform includes 3 types of tasks, the results are different according to the different types of tasks, described as the following:

- Training Task: a TensorBoard to monitor the training details, the log file.
- Evaluation Task: the evaluation results, including loss, accuracy, recall, precision, F1-score, and confusion metrics.
- Prediction Task: the prediction results that can be downloaded to the local path.

3.3 Technology Selection

The implementation of our project is divided into website development, NLP algorithm development, and database design.

We can choose from many development tools, frameworks, and packages for each part. After careful consideration and discussion, we propose to utilize Browser/Server structure, SpringBoot, Vue, and Mybatis for website development, Python library "Transformers" for NLP algorithm development, and Oracle for database design.

3.3.1 Browser/Server structure

The system mainly provides service to users with a certain understanding of NLP and have simple training model requirements or who want to learn NLP but are deterred by the preparatory work such as complex environment configuration. Therefore, the browser/server structure can fulfill the requirement.

1. The structure does not need to install a client, running directly in a Web browser, and is compatible with mainstream browsers.

2. B/S structure can be directly deployed on the Internet network to achieve some privileges to control the purpose of multi-client access and interaction mode.

3. Because there is no need to install the client, there is no problem with updating multiple clients and upgrading servers.

3.3.2 React + SpringBoot + MyBatis

React is a JavaScript library developed by Facebook for building user interfaces, mainly for building UI. Now available on Facebook and its Instagram apps. It differs from the massive AngularJS in that it only focuses on the V(View Layer) in

the MVC(Model View Controller) framework, which makes React easy to integrate with the developer's existing development stack.

React can be integrated without configuration, automatically manage dependencies and quickly build projects. Furthermore, the project can run independently without external containers and provide application monitoring at runtime, significantly improving development and deployment efficiency.

In the persistence layer, SpringBoot supports relational databases and non-relational databases. In Mybatis, on the one hand, queries are written in XML files for unified management and optimization, which decouples SQL queries from source code. On the other hand, it provides mapping labels, supports the mapping of object and database field relations, the mapping of object relation mapping labels, and the formation of object relations.

3.3.3 Python library "Transformers"

Python benefiting from its simplicity, consistency, flexibility, and great third-party libraries, is the most popular language in machine learning. The implementation of the algorithm part is developed by Python. We can efficiently process data and realize NLP algorithms with the help of Python and its deep learning toolkits.

We carefully tease widely-used deep learning frameworks and toolkits, such as sklearn, spacy, and Transformers. The deep learning framework provides basic operators for computations and accelerates relevant calculations efficiently, such as matrix multiplication. The deep learning toolkit decides what to compute. We adapt Pytorch[3] as our deep learning framework and Transformers as a toolkit to achieve our customized design of NLP models. Pytorch is an optimized dynamic

[3] https://pytorch.org/

graph framework to compute tensors by GPUs and CPUs, which is easy to learn and use. Transformers[4] is a popular Python library developed by Huggingface, which provides thousands of pre-trained models to perform tasks on different modalities such as text, vision, and audio. Transformers is not limited to deep learning frameworks as it seamlessly integrates three deep learning frameworks – Jax, Pytorch, and TensorFlow.

[4] https://huggingface.co/docs/transformers/index

4 System Analysis and Implementation

4.1 Front-end

4.1.1 Pack Common Components

Traditional web development based on HTML, js, and CSS requires tremendous coding. The logical processing file of a page covers hundreds or even thousands of lines of codes. Generally speaking, it is hard to read and maintain based on this kind of development mode, as once users want to add new features or functions based on existing codes, it may cause unexpectedly adverse effects. To solve this problem, we introduced the idea of packing common components while developing our platform.

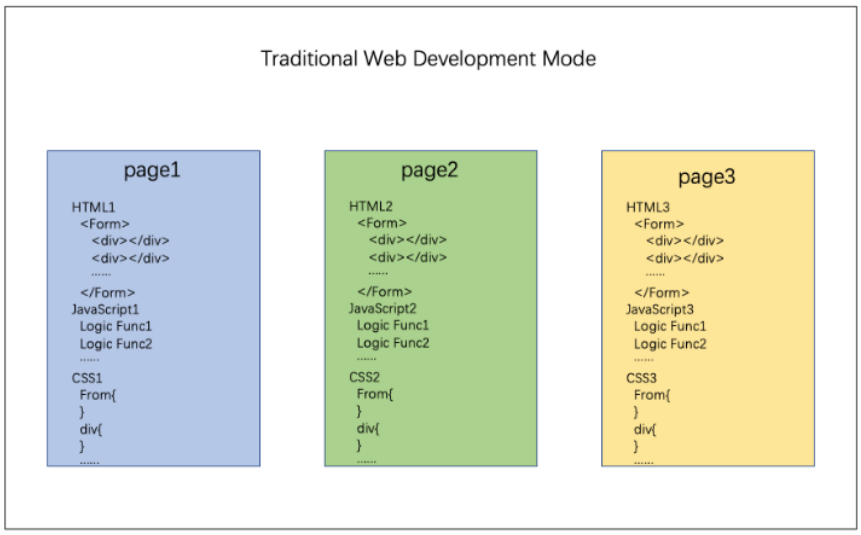


Figure 4: Traditional Web Development Mode

Front-end componentized development is to separate a certain part of the page. Moreover, encapsulate the data layer (M), view layer (V), and control layer (C) onto

one component in the form of a black box, exposing some out-of-the-box functions and properties for external pages to call.

A component contains CSS, JavaScript, template, style, and interaction between the components, which almost covers all the content of the component. Furthermore, it only needs to be called externally according to the properties, functions, and event processing set by the components. That is, the internal implementation logic of the component is not considered at all. To the outside, the component is a complete black box, which can be encapsulated in multiple layers by calling multiple small components, and finally encapsulated into a significant component for external calls.

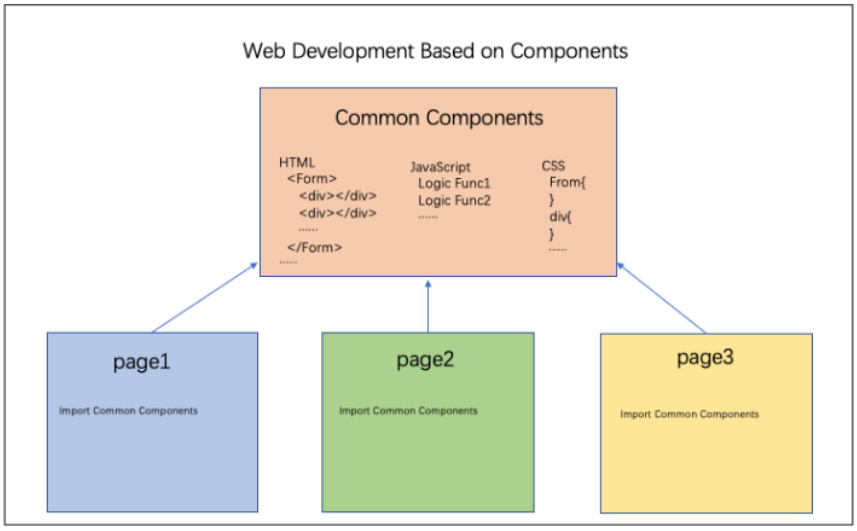


Figure 5: Web Development Based On Components

In our project, we packed various common components including Input box, Select drop-down boxes, Sliders and etc(shown in Figure 6, 7). And then, we can construct a form to wrap a layer on these components, which is a Form component, and it can

be used for transferring relative parameters.

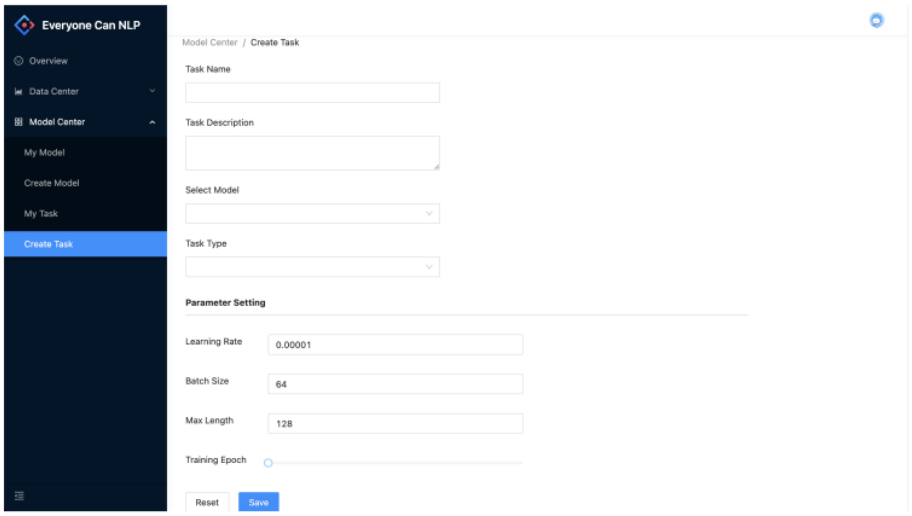


Figure 6: Create Task

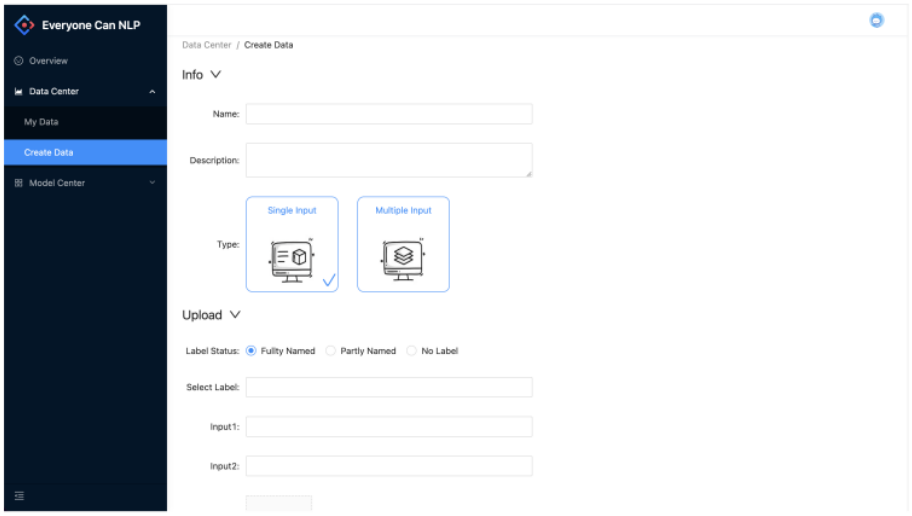


Figure 7: Create Data

As we entirely use componentized development, developing a page is like building blocks, splicing together various components and merging them. It is a complete system.

4.1.2 Data responsive interaction

In our platform, we support users to visualize the architecture of a specific model. The traditional solution is that the front-end designs the corresponding template for each model. Moreover, the front-end renders different templates according to the model chosen by users. It is evident that the process of designing and maintaining templates is complex and time-consuming. Besides, as users change the model more and more frequently, the web search engine will render more and more slowly. Therefore, we initially abstract the standard architecture of various models. Furthermore, specific parameters will change the content, including the number of layers corresponding to models.

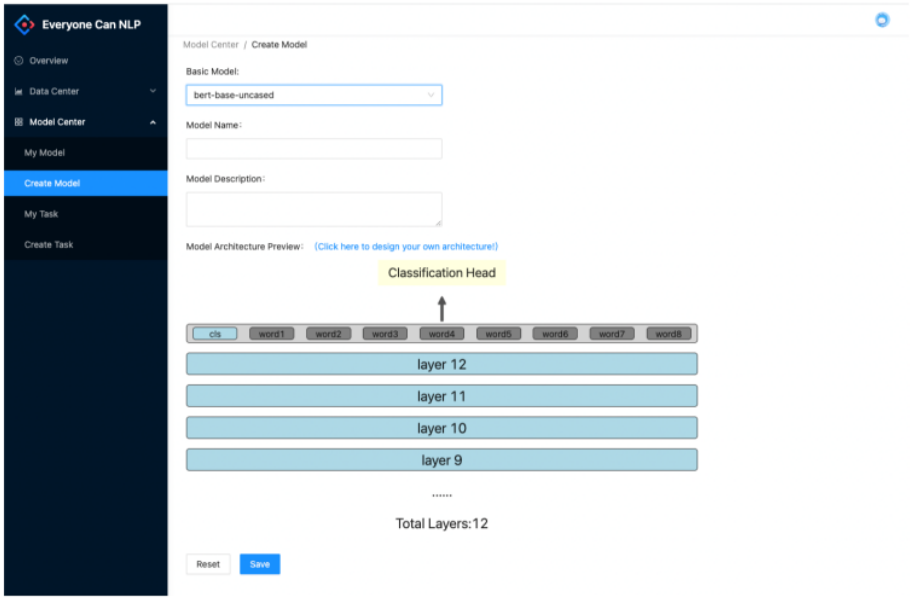


Figure 8: Create Model: bert-base-uncased

For real-time interaction of data and model structures, the front-end defines Data-View Conversion Functions and ultimately calls the conversion function to gen-

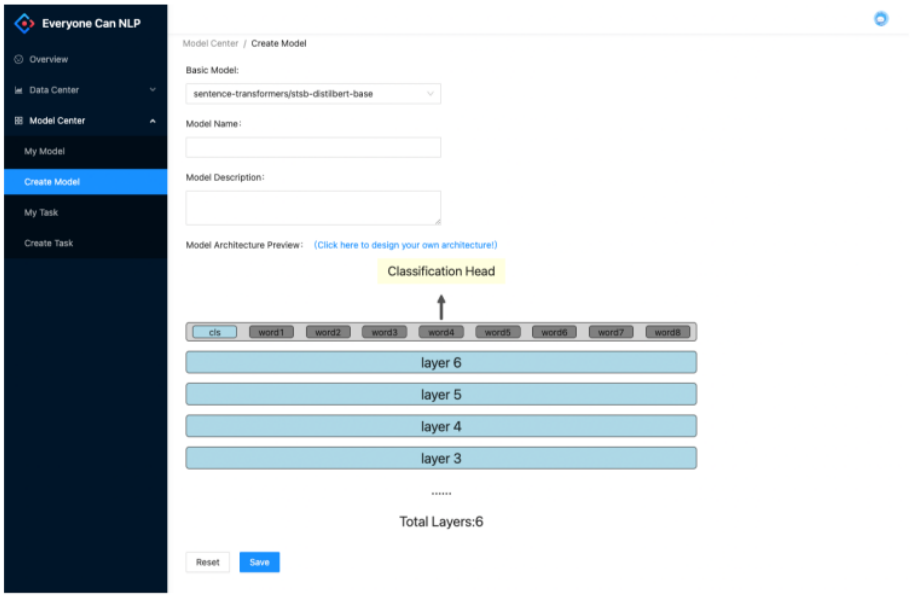


Figure 9: Create Model: sentence-transformers/stsb-distilbert-base

erate the initialization view. According to the mapping relationship of the same conversion function, the partial view update is automatically triggered. Finally, we constructed the data-to-view conversion (synchronization) functions specified only once, and the framework itself handles both initialization and update. Finally, the data and the view are synchronized at any time, and when the data changes, the view changes synchronously. Developers only relate to the data source, not how the views are updated.

We also support users in designing their architecture for each layer of the specific model. The basic implementation idea is similar to that of visualizing model architecture. Moreover, as layer architecture is more complex than traditional architecture, we provide more customized parameters, including the choice of activation function, classifier, etc.

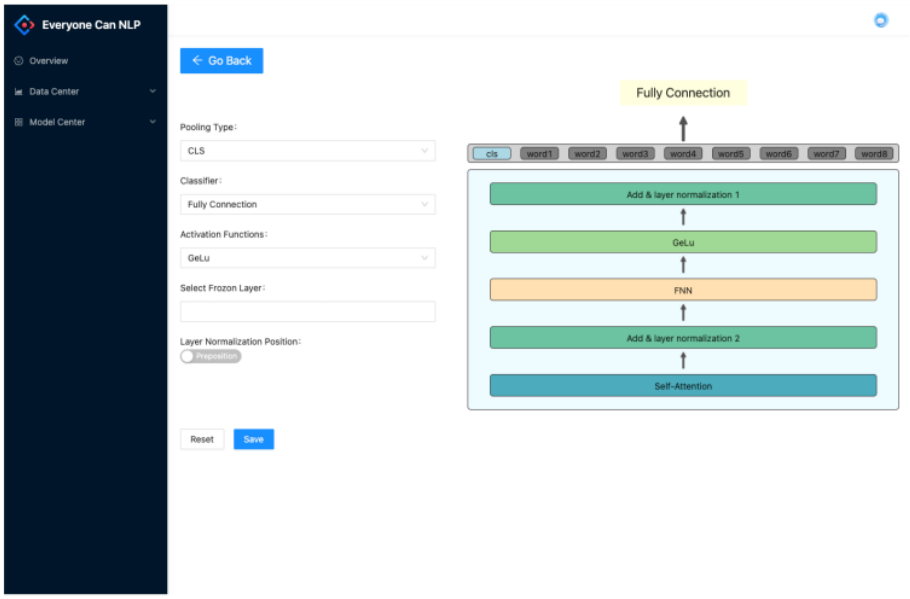


Figure 10: Create Model: Visualizing model architecture

4.1.3 Visualizing Graphs

The front-end displays a bunch of visualizing graphs, including Violin, Pie, and Word Cloud charts, which aim to help users understand the dataset better. To implement these graphs, we introduce Ant Design Chart[5], a lightweight javascript graphics library, pure js implementation, MVC framework, and data-driven. And the Ant Design Chart can be easily implemented in the project by installing the corresponding npm package named @ant-design/charts.

Nevertheless, the visualizing dataset size is generally huge, which means the speed of the rendering page is slow. To solve this problem, we mainly optimize from two aspects, including utilizing web storage to accelerate loading data speed and utilizing a particular hook to avoid re-rendering. Initially, we fetched the data information while users firstly entered the Data Overview page. And then, we utilize the

[5]https://charts.ant.design/

Figure 11: Data Graphs

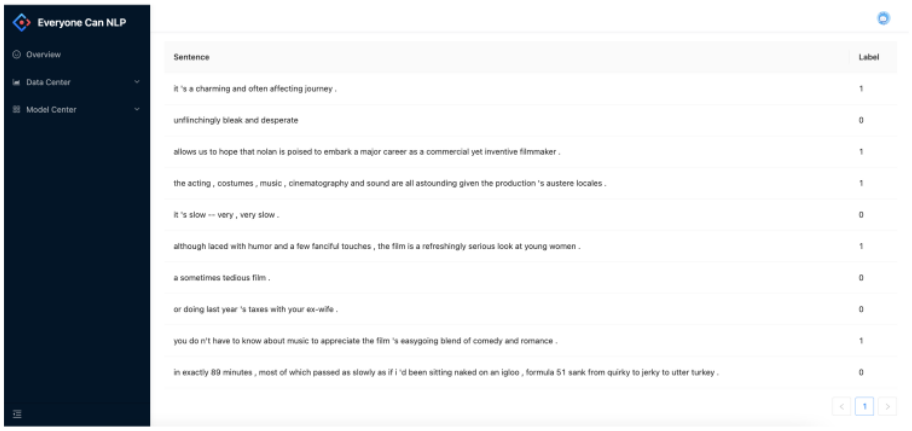


Figure 12: Data Content Table

LocalStorage API to store the data information.

Therefore, when users enter the page again, our page will fetch the data from web storage instead of fetching through an HTTP request.Moreover, the components containing statistic charts will also be repeat rendered even if the corresponding data information does not change, as react will repeatedly render the page while the data in useState that does not belong to the chart component is modified in other isolated components. Therefore, we imported a particular hook named useMemo to avoid re-rendering. useMemo is aimed at the pain point that the execution period of useEffect is executed after the page is rendered.

4.1.4 Reverse Proxy

A reverse proxy server acts as a gateway to a web server. When the users send a request to a web server using a reverse proxy, users' requests do not go to the web server, and they go to the reverse proxy, which then determines if the request should be sent to the web server.

By introducing reverse proxies, requests do not reach the web server directly. They

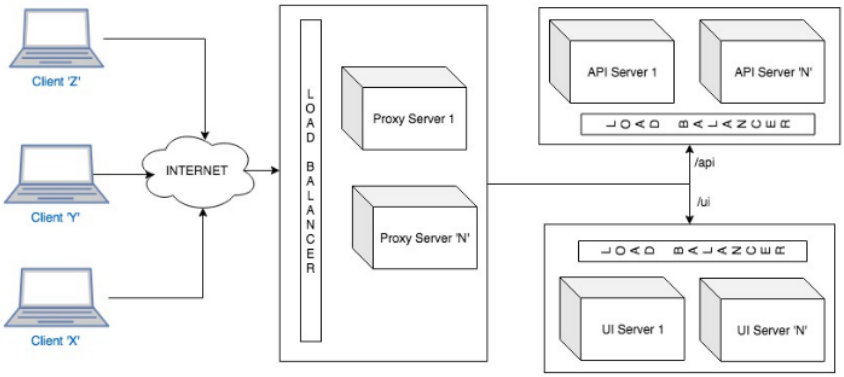


Figure 13: The principle of Reverse Proxy

help protect web servers from known vulnerabilities. Moreover, some reverse prox-

ies can also act as caching machines. Typically, if users send a thousand requests for the same resource in a minute, those requests will repeatedly reach the server requesting the same resource, wasting resources and precious time. When a resource is requested for the first time, the reverse proxy can cache (save) the obtained resource, and when the same resource is needed next time, it can be obtained directly, saving resources and time. The most significant advantage of a reverse proxy is that it can be regarded as a load balancer. High-traffic websites often face the problem of processing many requests per minute, slowing down their system performance, and destroying response times—using a reverse proxy guarantees a better user experience and faster response.

In our system, we use Nginx as a reverse proxy and deploy it on Alibaba Cloud server.

4.1.5 Consistent User Interface Style

Our front-end ensures UI consistency by automatically adapting page content to screens of different sizes. Since our platform is mainly accessed through the web, we mainly consider the adaptation work on the Personal Computer(PC) side in the process of project design. The primary solution is that with the ability of JavaScript to control the viewport, use rem to simulate the characteristics of the zoom effect according to view window size to achieve a set of solutions for adaptation purposes. Rem is a computed property value that is computed relative to the font size of the HTML element.By setting the value of the fontSize property of documentElement, the layout standard of the entire page can be unified. In addition, the front end also sets the width of the viewport to device-width and changes the default width of

the browser viewport (layout viewport and visual viewport) to the ideal viewport width so that the user can see the complete viewport in the ideal viewport. The contents of the layout viewport will also be viewed clearly. By setting the initial-scale, maximum-scale, and minimum-scale values of the viewport in equal proportions, we achieve 1 physical pixel = 1 css pixel, to adapt to the display effect of the high-magnification screen.

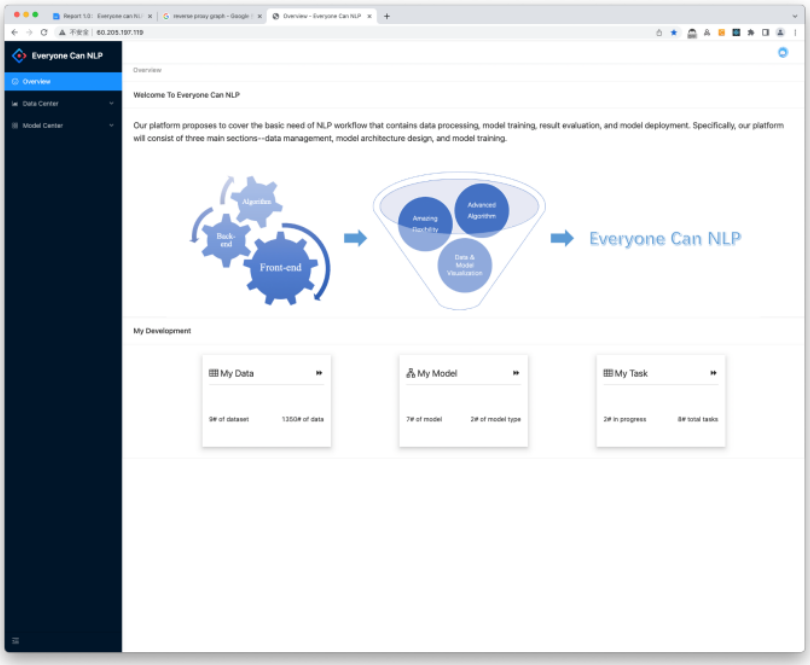


Figure 14: The effect on Big PC

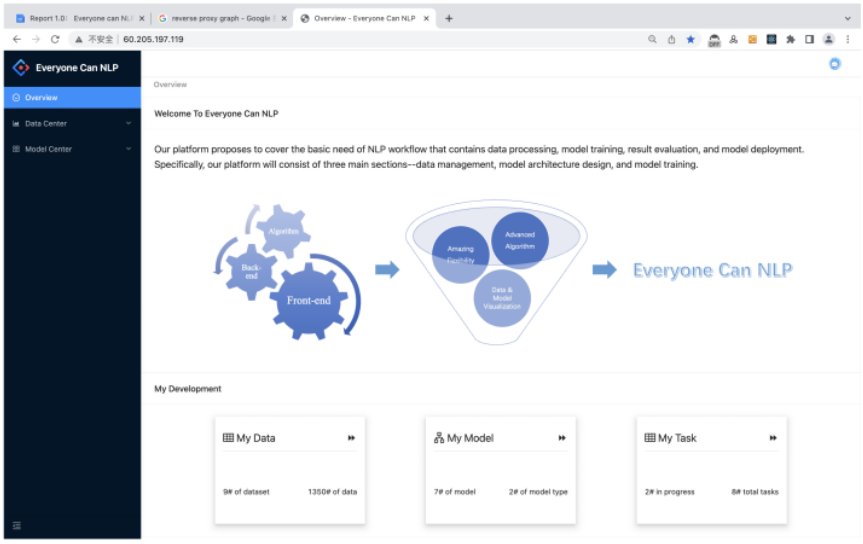


Figure 15: The effect on Small PC

4.2 Back-end

4.2.1 Security of Web applications

In our platform, Web application security is embodied in user authentication, as shown in Figure 16. Authentication is to verify that a user is a legitimate principal in the system (i.e., user can log in to the system or not). User authentication generally requires a user name and password and verifies them to complete the process. Generally speaking, it is whether the system thinks the user has permission to access the system.

The user enters the username and password on the browser interface and then enters a randomly generated captcha with numbers and letters. In this step, the user's password is encrypted at the back-end during each login verification process and compared with the data in the database. This method of user authentication improves the overall security of the system. Once the problem of database information leakage occurs, it can also protect the privacy of users. After the background verification is passed, a JWT (JSON web token) and authorization code will be returned. In subsequent requests, the authorization key must be included in the request header.

For some static resources, such as image files, login interface, etc., setting a whitelist in WebSecurityConfigurerAdapter can complete the user's access requirement without logging in.

4.2.2 High-speed Buffer Cache

Use the Redis cache as a data storage media in scenarios requiring frequent changes or set variable expiration policies. The efficiency of reading and writing data in Re-

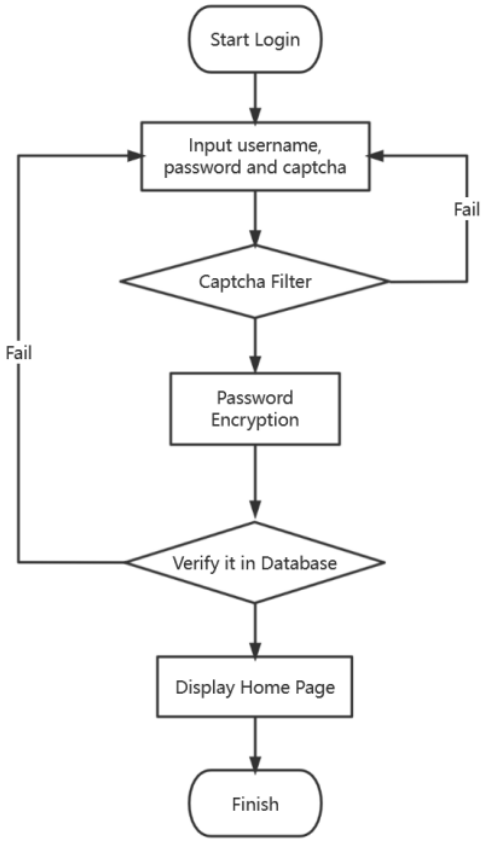


Figure 16: The Process of User Authentication

dis is extremely high, far exceeding that of databases. At the same time, the data stored in Redis is persistent, and the data will not be lost after a power failure or restart. Redis's storage consists of three parts: memory storage, disk storage, and log files. After restarting, Redis can reload data from the disk into memory, which can be configured through configuration files. The data stored in the cache often needs to be fetched frequently, so the I/O of the system directly reading the disk to obtain the data can be greatly saved. And more importantly, the speed of system response can be significantly improved. In this system, the cache is used in functional modules such as verifying the verification code when logging in, uploading data files, and accessing tensorboard ports for different training tasks.

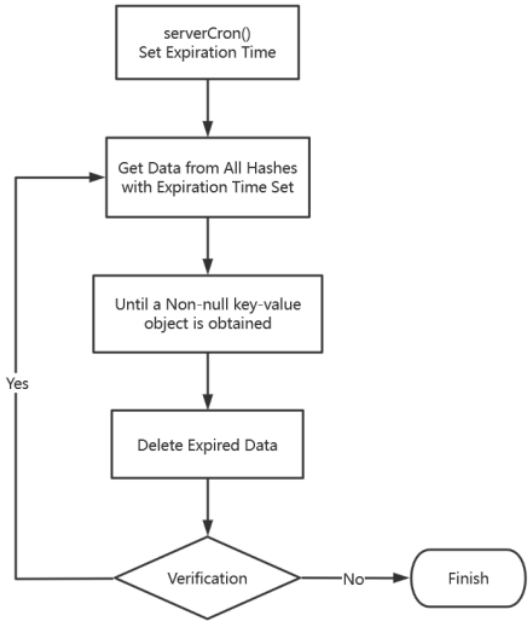


Figure 17: Flowchart of Expired Object Destruction

During the development process, use the packaged redisUtil to create new key-value objects and set the expiration time. For requests from different interfaces,

there is no need to use communication to pass values or to persist in the database. Objects can be automatically destroyed after the set period is exceeded. The following flow chart shows the destruction strategy of the cache.

4.2.3 Application Programming Interface

The back-end service provides a unified access portal for the front-end (usually, a single service is deployed). It provides HTTP services for the front-end to invoke in a restful style. The external unified API returns the required data based on the requested operation data. It achieves data transmission and absolute decoupling of the front-end and back-end services. For all data management operations, the front-end only needs to send different requests to the back-end interface, which processes, calculates, and filters the obtained data and packages the data in JSON format for parsing.

In fact, when developers were developing with the SpringMVC+Spring+Mybatis framework, building a Java project can be a hassle, with so many configuration files to build and conflicting dependencies on open source components that it can take days to build a project. SpringBoot is used in the project as a back-end framework that enables AOP (Aspect Oriented Programming) and DI (Dependency Injection). Setting up a Web back-end project is easy and convenient. It saves not only the time to set up a project but also the time to set up a development environment. Springboot is built with Tomcat middleware, enabling one-stop deployment without the need for developers to manually build it. For the entire team development and maintenance, this development approach is a significant cost saving. The framework automatically annotates objects, making the code more readable to

developers, testers, and maintainers. Thereby, the back-end services are deployed on the cloud server, which can easily tackle the problem of CORS (cross-Origin Resource Sharing) and achieve separate development of front and back ends.

The get interface corresponds to the query function of different entities. In the platform, the dataset and model support the equivalent query of id and the fuzzy query of name, and the task can perform status classification and fuzzy query of the name.

Function	Get Dataset Interface
Path	/api/v1.0/dataset/get
Request Type	GET
Parameters	{id, name}
Response Body	Result{code:0, msg:"success", List.Dataset{}}
	Code:204 No Content
	Code:401 Unauthorized

Table 1: Get Dataset Interface

The following table shows the API for creating a model. For the API to add an entity, it needs to be a POST request. The request body needs to contain the information required to create a new model, which is sent by the client and parsed through the RequestBody annotation. After this object is validated and the parameter is completed, it will persist in the database.

Function	Create Model Interface
Path	/api/v1.0/model/add
Request Type	POST
Body	{Model{}, config: ModelConfig{}}
Response Body	Code:200 Result{code:0, msg:"success", List.Model{}}
	Code:204 No Content
	Code:401 Unauthorized

Table 2: Create Model Interface

In order to deletes an object, the server only needs to find the corresponding row in the database by id. In the relational database, the primary key index is automatically generated after a table is created, so using the primary key to query data is the fastest way. As shown in the table below, only one parameter, task_id, is required to delete the corresponding task.

Function	Delete Task Interface
Path	/api/v1.0/task/del/{task_id}
Request Type	DELETE
Parameters	{task_id}

Response Body	Code:200 Result{code:0, msg:"success"}
	Code:204 No Content
	Code:401 Unauthorized

Table 3: Delete Task Interface

Updating object information is similar to creating. However, it should be noted that the update operation requires an additional object ID. In the example given in the table, the id is model_id. Because the model_id is generated by the server when created, it does not need to be sent by the client. Nevertheless, when an entity is updated, it needs to be found and updated according to the id. Therefore, the object's id is a non-null value that needs to be placed in RequestBody. The PUT request is used here. Because in the definition, PUT is an idempotent operation, it is known which specific object is being operated. Therefore, using the PUT operation here is reasonable, which is in line with the design principles of REST API.

Function	Update Model Interface
Path	/api/v1.0/model/update
Request Type	PUT

Body	{
	Model{},
	”config”: {
	”activation”: ”string”,
	”clsType”: ”string”,
	”freeze”: [],
	”lnType”: ”string”,
	”lyrType”: ”string”,
	”poolerType”: ”string”
	}
	}

Response Body	Code:200 Result{code:0, msg:”success”}
	Code:204 No Content
	Code:401 Unauthorized

Table 4: Update Model Interface

4.2.4 Database Design

The user, dataset, model, and task information is stored in the database. The persistent storage of data uses MySQL, which is a relational database. And there may also be some dependencies between the tables of the schema. For instance, the one-

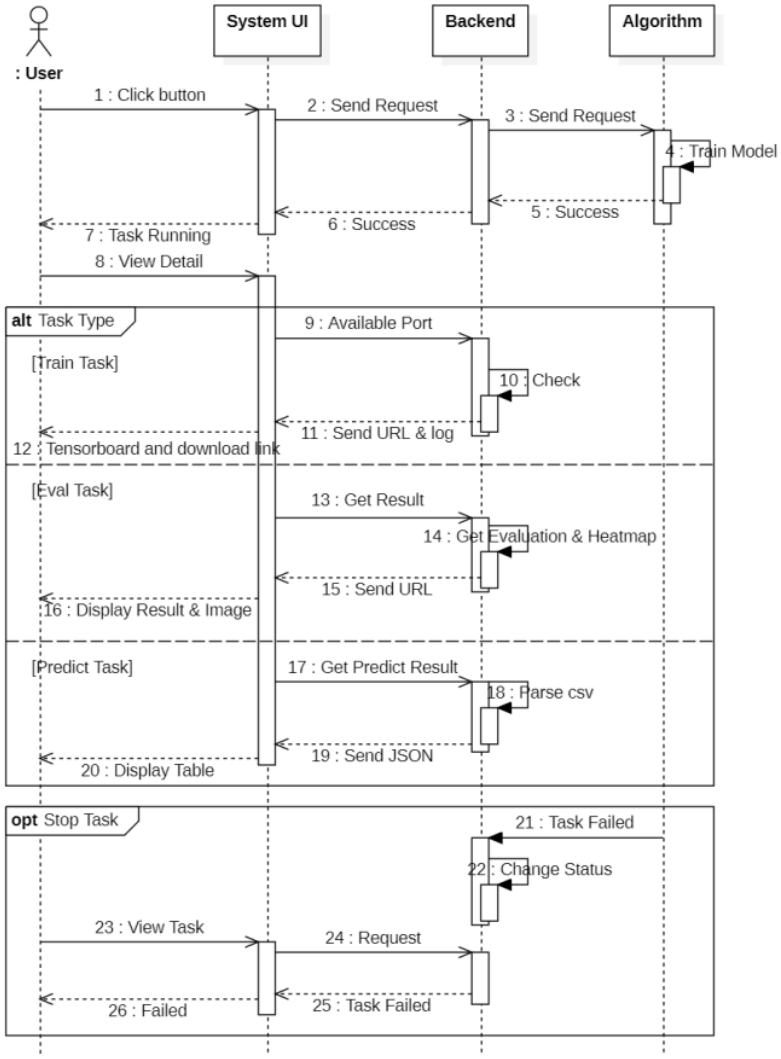


Figure 18: Sequence Diagram of Task Execution

to-one relationship between models and model configurations, the one-to-many relationship between tasks and datasets, etc. .

The User table is used to store user information for system authorization. The user id and username cannot be null and unique, and the password field stores the encrypted user password. The primary key of the user table is the user id, and there is an index for querying.

Column	Type	Feature	Note
id	BIGINT	NOT NULL AUTO INCREMENT	User ID
email	VARCHAR(64)	DEFAULT NULL	User Email
password	VARCHAR(64)	NOT NULL	User Password
username	VARCHAR(64)	NOT NULL	Username

Table 5: User Table

Model and model-config table respectively store the model and its configuration persistently. After the task runs, the parameters in the table will be parsed into a configuration JSON file for model training.

The foreign key of the model-config table, model_id, which references the model table, is also the primary of it. If users create, update or delete the model entity, the corresponding model-config will be cascaded to create, update and delete. In the sense of similarity, the relationship between model and model-config is one-to-one, which can be understood as the composition relationship in OOM (Object-Oriented Modeling). It means that they have the same life cycle.

Column	Type	Feature	Note
model_id	BIGINT	NOT NULL AUTO INCREMENT	Model ID
model_name	VARCHAR(45)	DEFAULT NULL	Model Name
basic_model	VARCHAR(45)	DEFAULT NULL	Basic Model, select in dropdown box
description	VARCHAR(255)	DEFAULT NULL	Model Description
update_time	DATETIME	AUTO GENERATE	Model Update Time
user_id	BIGINT	FOREIGN KEY(User.id)	User ID

Table 6: Model Table

Column	Type	Feature	Note
activation	VARCHAR(45)	DEFAULT Gelu	Activation Functions
cls_type	VARCHAR(45)	DEFAULT FC	Classifier
freeze	VARCHAR(45)	DEFAULT NULL	Freeze Layer
ln_type	VARCHAR(255)	DEFAULT pre	Layer Normalization Position
lyr_type	VARCHAR(45)	DEFAULT NULL	Layer Type
model_id	BIGINT	FOREIGN KEY(Model.id)	Model ID
pooler_type	VARCHAR(45)	DEFAULT cls	Pooling Type

Table 7: Modelconfig Table

There are two types of datasets. Only label and input1 are needed for a single-

input dataset, while for the multi-input dataset, input2 is required. Label and input information is used in data analysis and task configuration JSON file generation.

Column	Type	Feature	Note
dataset_id	BIGINT	NOT NULL	Dataset ID
description	VARCHAR(255)	DEFAULT NULL	Dataset Description
input1	VARCHAR(45)	DEFAULT 1	Input1
input2	VARCHAR(45)	DEFAULT 2	Input2
label	VARCHAR(45)	DEFAULT 0	Label
name	VARCHAR(45)	DEFAULT NULL	Dataset Name
path	VARCHAR(255)	DEFAULT NULL	Path
status	INT	DEFAULT 0	Status
type	INT	DEFAULT 0	Type
update_time	DATETIME	AUTO GENERATE	Update Time
user_id	BIGINT	User ID	User ID

Table 8: Dataset Table

The task table is adapted for three types of tasks. The training task has a dataset for test and training. Evaluating or predicting tasks are based on a training task and a test dataset. And some fields have default values, that is, the initial parameters recommended.

Column	Type	Feature	Note
task_id	BIGINT	NOT NULL	Task ID
task_name	VARCHAR(45)	DEFAULT NULL	Task Name

task_type	INT	NOT NULL AUTO GENERATE	Task Type
batch_size	INT	DEFAULT 64	Batch Size
dataset_id_test	BIGINT	FOREIGN KEY(Dataset.id), NOT NULL	Test Dataset ID
dataset_id_train	BIGINT	FOREIGN KEY(Dataset.id), NOT NULL	Train Dataset ID
dataset_id_train_task	BIGINT	FOREIGN KEY(Task.id), NOT NULL	Train Task ID
description	VARCHAR(255)	DEFAULT NULL	Task Description
start_time	DATETIME	AUTO GENERATE	START Time
end_time	DATETIME	AUTO GENERATE	END Time
epoch	INT	DEFAULT 128	Epoch
learning_rate	VARCHAR(45)	DEFAULT 1e-5	Learning Rate
max_length	INT	DEFAULT 128	Max Length
model_id	BIGINT	FOREIGN KEY(Model.id)	Model ID
status	INT	DEFAULT 0	Task Status
user_id	BIGINT	FOREIGN KEY(User.id)	User ID

Table 9: Task Table

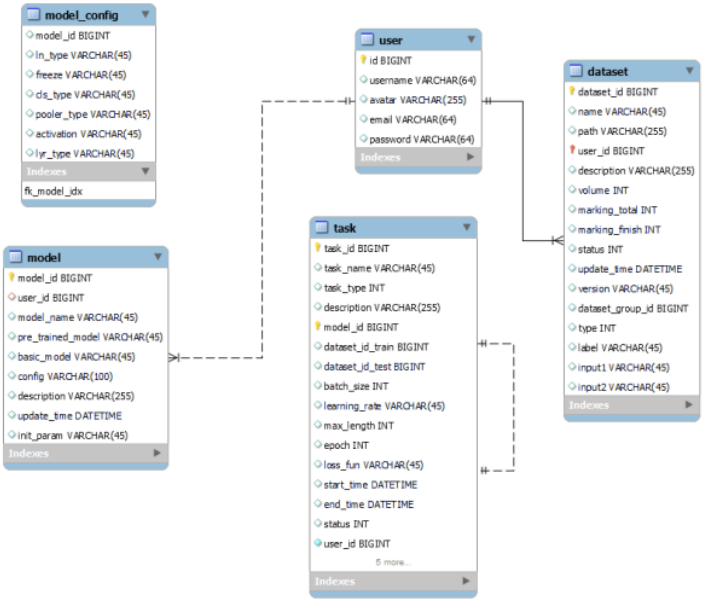


Figure 19: EER Diagram of Schema

4.3 Algorithm

Sentences are logical combinations of words based on linguistics. Unlike images, textual data are unstructured and vary from language to language. Humans can easily understand the semantic meaning of sentences by reading them. However, it is challenging for computers to process and comprehend. Learning representations of the meaning without losing too much information in sentences has been a classic and popular problem since the beginning of Natural Language Processing.

Text classification can be regarded as a process of understanding the semantic meaning of a sentence and classifying them into one class. The whole model can be

divided into a representation extractor and a classifier. An informative sentence representation, also called sentence embedding, can significantly enhance the performance of the classifier. Generally speaking, the representation extractor is a complex model that translates the sentence into an embedding making the embedding easy to classify, and the classifier is a small-scale model predicting the class. Our platform selects BERT [7] as the representation extractor and provides many kinds of classifiers, including Feedforward Neural Network (FNN), TextCNN[6], and attention mechanism [8].

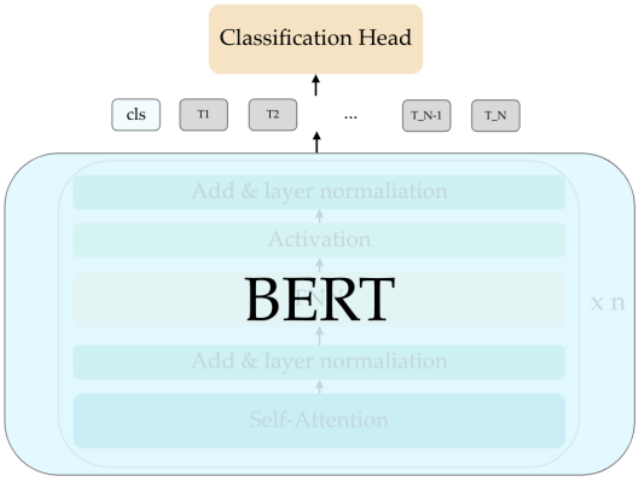


Figure 20: Overview of model

4.3.1 Overview of BERT

BERT is a milestone large-scale pre-trained language model with multiple layers of Transformer [9] encoders. BERT has been studied extensively since it achieves start-of-the-art performance in 11 NLP tasks like GLEU. BERT outperforms traditional algorithms because of its innovative architecture, model scale, adequate unlabeled data, and plentiful pre-training tasks. BERT has 110 million parameters and con-

ducts the pre-training on 3300 million unlabeled words corpus.

The pre-training tasks of BERT consist of the masked language model task and the next sentence prediction. A Language model is to compute the probability of the next word given a word composition, which can only consider the leftward words, while a masked language model predicts the masked word within a sentence considering both sides of the sentence. Therefore, BERT has a more global view of sentences and integrates bidirectional information into sentence embedding. BERT grasps the relationship between sentences by the next sentence prediction task that identifies whether two sentences come from the same passage. These pre-training tasks qualify BERT with powerful language understanding ability and thus advanced classification ability.

The basic-size BERT with 12 layers of encoder transfers readable words into contextual embeddings. Each encoder layer comprises multi-head self-attention layers, layer normalization, and FNN, presented in Figure 21. The multi-head self-attention layer is the most significant part of BERT. It integrates dependencies between words into embeddings through the attention mechanism and flows embeddings into layer normalization. Layer normalization serves to regularize and smooth the distribution of embeddings to accelerate the convergence of training. FNN and activation functions enhance the complexity and nonlinearity of the model. After computations of BERT, the first word embedding, named [CLS] shown in Figure 20, representing the whole sentence is fed into the classification head to predict the final class.

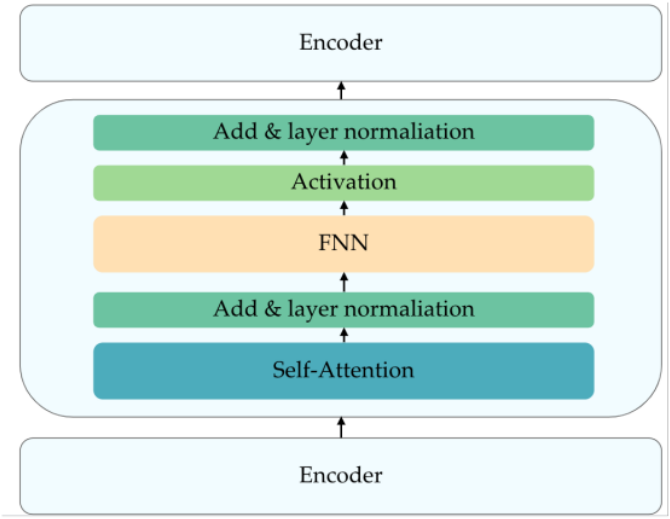


Figure 21: Details of BERT encoder

4.3.2 Model Optimizations

After the debut of BERT, there has been a spate of interest in modifying the architecture of BERT attempting to achieve a better performance. Although BERT achieved start-of-the-art performance at that time, it still can be improved. We integrate some effective modifications of model from different aspects in recent researches into our platform.

1. Add & layer normalization layer

We incorporate Pre-LN into our platform for users to change model architecture to achieve better performance.

After the computation of multi-head self-attention layer and FNN layer, there is a residual connection and layer normalization layer. On the one hand, the residual connection [10] eazes the gredient vanishing problem and enables a deeper model. On the other hand, the residual connection tends to amplify the variance of embeddings. Layer normalization acts to re-scale embeddings' distribution. How-

ever, layer normalization weakens the effect of residual connection requiring a warm-up stage and smaller learning rate for model convergence. Therefore, Pre-LN[11, 12, 13, 14] is proposed to facilitate training without strict initialization and the warm-up stage.

$$Pre\ Norm : x_{t+1} = x_t + F_t(Norm(x_t)) \tag{1}$$

$$Post\ Norm : x_{t+1} = Norm(x_t + F_t(x_t)) \tag{2}$$

2. Activation function

Our platform supports not only the original activation function GELU but also abundant activation functions, inclduing ReLU, squared ReLU, and SiLU.

Activation functions provide nonlinearity to the FNN layer. A satisfying activation function should be easy to compute, zero-centered, and avoid gradient vanishing problems. The original activation function in BERT is GELU [15] that randomly regularizes the activation value. Primer [16] experiments other activation functions such as SwiGLU [17], Swish [18] and finds squared ReLU can efficiently and maximumly improve the performance of BERT. Squared ReLU is a simple and easy-implemented modification of ReLU, which squares the output of ReLU, demonstrated as Figure 22.

3. Pooling method

BERT as a representation extractor encodes every token into token embedding and regards the first token named [CLS] as the flag of the entire sentence sequence. There is an MLP layer over [CLS] in BERT's original implementation. The flag flows into the classification head to compute the probability of classes. However, there are

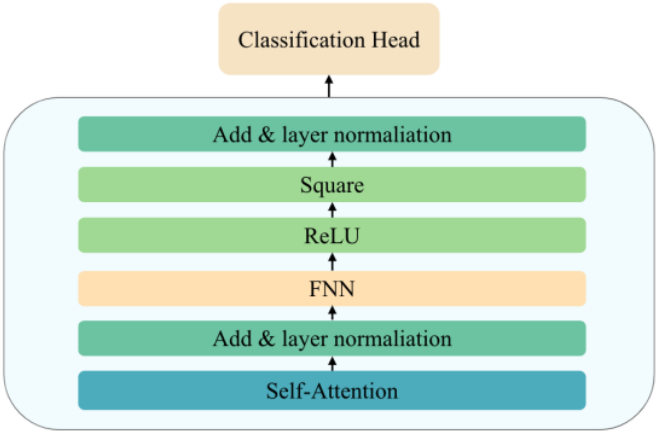


Figure 22: The illustration of squared ReLU

many kinds of pooling methods [**?**] to group all processed token embeddings into a sentence embedding. The semantic meaning of a sentence may be difficult to encompass solely by [CLS] representation. [**?**, 19, 20] indicate that better performance can be obtained using the average token embedding of the pre-trained model. Our platform has a wider variety of pooling approaches than [CLS], including [CLS] before the MLP layer, the average token embedding of the last layer, the average token embedding of the last two layers, and the average token embedding of the first and last layers.

4. Classification head

The classification head classifies the representation of a sentence, i.e., sentence embedding into one class. The structure of the classifier has a significant impact on how much information can be leveraged inside the extracted features. Our platform provides FNN, TextCNN [6], and attention mechanism [8] as classification head and allows users to design the satisfying classifier.

5. Frozen representation extractor

We generally leverage a pre-trained model and fine-tune it to fit downstream tasks.

Adaptation to downstream tasks may lead to the collapse of the existing parameters and the loss of the powerful understanding ability gained from the massive amount of pre-trained data. Therefore, our platform enables users to freeze model weights to avoid catastrophic forgetting.

6. Pre-trained parameter

Our platform supports initializing the model from pre-trained weights registering in the HuggingFace models hub. We provide the most popular encoder-based model weights from the model type, the model scale, and supported languages. All pre-trained parameters are described in Table 10. It is worth noting that the characteristics of each pre-trained parameter are meticulously labeled beside each name and implemented by the front-end, illustrated in Figure 23.

Basic Model	Pre-trained Parameter
BERT	Pre-trained Parameter
	bert-base-uncased
	bert-base-cased
	bert-large-uncased
	hfl/chinese-macbert-base
	bert-base-chinese
	bert-base-multilingual-cased
	SpanBERT/spanbert-large-cased
	sentence-transformers/all-MiniLM-L6-v2
	sentence-transformers/paraphrase-MiniLM-L6-v2
	DeepPavlov/rubert-base-cased-conversational
DistillBERT	sentence-transformers/stsb-distilbert-base

	distilbert-base-uncased
	distilbert-base-uncased-finetuned-sst-2-english
	distilbert-base-multilingual-cased
RoBERTa	roberta-base
	roberta-large
	xlm-roberta-base
	xlm-roberta-large-finetuned-conll03-english
	cardiffnlp/twitter-xlm-roberta-base-sentiment
	cardiffnlp/twitter-roberta-base-sentiment
ALBERT	albert-base-v2

Table 10: Pre-trained Parameter Supported By The Platform

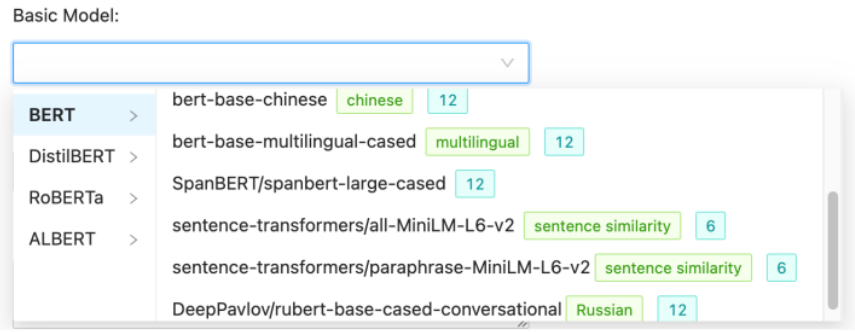


Figure 23: Characteristic of pre-trained parameters shown in our website

4.4 Tripartite System Corporation

The services of the algorithm part are deployed by Ray Serve[6], which provides APIs to the back-end. The algorithm services include training task, evaluation task, prediction task, and TensorBoard setup, described in Table 11. The back-end can start

[6]https://docs.ray.io/en/latest/serve/index.html

or shut down services through corresponding APIs. The algorithm APIs require the storing path of the task's configuration and task's ID. Once the task finishes or gets errors, the algorithm APIs will send the message with the task's ID to the back-end and display the status of the task to users. In addition, the output of the task, including the task's log, and model's parameter, is saved to the path appointed by the back-end in the task's configuration and shown on the website.

Function	Start training/evaluation/prediction task	Stop training/evaluation/prediction task	Start TensorBoard
Path	train/evaluate /predict	stop_train /stop_evaluate /stop_predict	tensorboard
Request Type	POST	POST	POST
Body	{config_path: str, user_dir: str, task_id: str}	{user_dir: str}	{user_dir: str, port: str}

Table 11: Details of the algorithm APIs

For the back-end, after receiving the front-end request, the server needs to first verify the user information. And then, parse the object according to the type of task to generate the JSON file needed to train the model, which will be persisted in the server. Finally, PostUtil utility class will package HTTP requests and send them to the algorithm side, with the path to the configuration file and the task id.

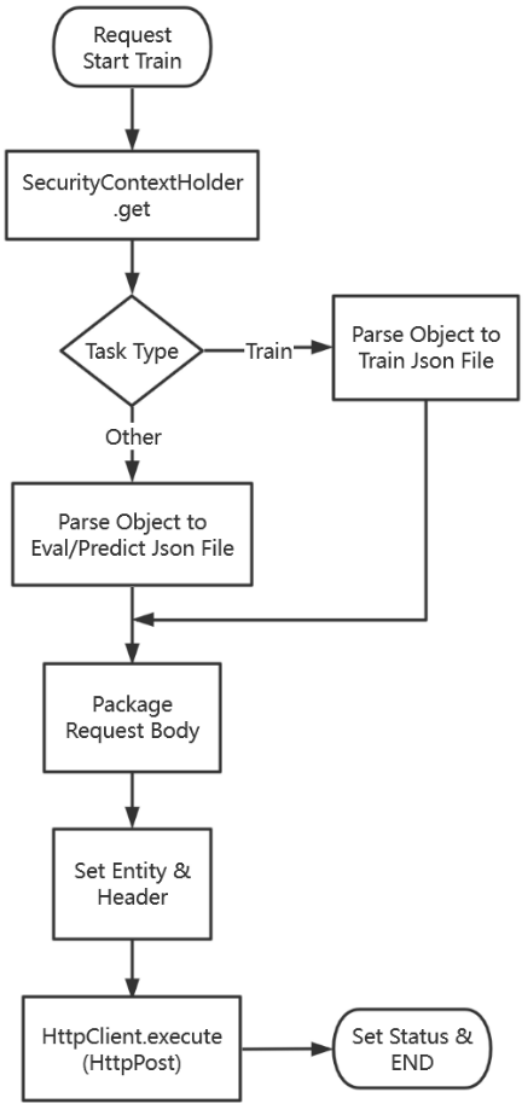


Figure 24: Flowchart of Sending Start Training Command in Server

5 Discussions

Our platform integrates the state of art algorithms to optimize model classification performance. In this section, we carry out great experiments to discuss the contribution and significance of our provided modifications to the model. We systematically study their effectiveness in STS-2[7] dataset by selecting the original BERT as a baseline and altering one aspect of the setup at a time.

SST-2 dataset has 11,855 single sentences extracted from movie reviews labeled by positive sentiment or negative sentiment. We gather the baseline results from SST-2 dataset leaderboard[8] and also train our own baseline result. Our baseline model uses the naive BERT with "bert-base-uncase" as the initial checkpoint and a fully-connection layer as the classification head.

We train the model with a batch size of 128, a maximum sequence length of 32 for ten epochs, and select the checkpoint with greater accuracy by evaluating the performance in the validation set. We utilize AdamW with learning rate of 5e-5, learning rate with warmup ratio of 0.1, and linear decay of learning rate.

We alter one improvement at a time and compare it with the baseline results, shown in Table 12. We use accuracy as the evaluation metrics since the SST-2 dataset is balanced towards labels.

Ablation	Modification	Accuracy	Diffience
Baseline	leaderboard	91.2	
	ours	91.4	0.2
layer normalization	Pre-LN	92.3	1.1

[7] https://huggingface.co/datasets/sst2
[8] https://paperswithcode.com/sota/sentiment-analysis-on-sst-2-binary

activation	squared ReLU	92.2	1
pooling method	cls_before_pooler	92.1	0.9
	avg	91.2	0
	avg_top2	91.7	0.5
	avg_first_last	91.2	0
classifier	attention	92.3	1.1
	TextCNN	92.2	1
freeze	1,2,3	91.8	0.6

Table 12: SST-2 results. Ablation over different modifications. One aspect is changed at a time from the basic model BERT_BASE. The Difference column demonstrates the improvement compared with the first row.

The experiments illustrate that the modifications provided by our platform effectively strengthen model performance. The baseline result from leaderboard is 91.2, while our baseline result is 91.4 since we use AdamW as an optimizer rather than Adam. Similar to the conclusion of previous research [16, 11, 12, 13, 14], squared ReLU and Pre-LN play a crucial role in improvement, reaching 92.2 and 92.3 respectively. Moreover, the classification ability also has a significant influence on performance. However, the pooling method of word embeddings shows minor improvement in our experiments. The result of the freezing parameter corroborates that the lower layers of BERT extract basic information which can be frozen during fine-tuning.

6 Conclusions

The platform to build natural language processing models has gained widespread interest in the industry. It lowers the barriers for users to design and use natural language processing models.

Our team implements a well-visualized and user-friendly NLP model building platform in the project. The platform consists of three functional modules, which complete the whole process of model building from data upload, and model design to model output result and adopt the graphical design to reduce the requirement of the user's coding ability. The platform has two core innovations - one is to make the model structure editable in the front-end through the canvas, allowing users to design deeper into the model layers; the other is to make the NLP algorithm more flexible to match what user needs with higher freedom.

At the same time, the team also evaluated the effect of the models implemented on the platform. As a result, the experiment proves that the platform can help users build models with high accuracy and achieve the Sota algorithm, which can be applied to real-life problems and have good results.

7 Future works

Since the platform aims to be well-visualized and user-friendly, the team needs to improve the platform in both ways.

1. Provide more options in model design

Though we have offered many options to improve the freedom of how the user designs the model, loss functions like focal loss and label smoothing can be added to achieve better results.

2. Improve the rendering speed

The first page of the website is relatively slow to load The current project involves too many rendering projects webpack packaged with large modules. The first page is slow to render. The future can be dynamically introduced modules packaged in blocks to speed up the first screen rendering speed.

3. Add more data visualization to the result page

Due to server limitations, the front-end rendering is limited, so some of the data visualization of the results is abandoned, which is a great pity. If the server performance can be improved, users will get a better.

4. Refine the website layout

The current website is most fitted for the tiny PC screen and is not ideal with the adaptation of a large PC screen layout. Improvements can be made to make it more robust.

8 Distribution

Task items	PIC
Requirement Analysis	REN
High Level Design	CAI
Low Level Design	FAN, XU
Algorithm	CAI
Front-end	FAN
Back-end	XU
Data Analysis	REN
Unit Testing	FAN, XU
Integration Testing	CAI, REN
Report	ALL

Table 13: Task Table

References

[1] K. S. Jones, "What is the role of nlp in text retrieval?," in *Natural language information retrieval*, pp. 1–24, Springer, 1999.

[2] M. Hanussek, M. Blohm, and M. Kintz, "Can automl outperform humans? an evaluation on popular openml datasets using automl benchmark," in *2020 2nd International Conference on Artificial Intelligence, Robotics and Control*, pp. 29–32, 2020.

[3] J.-H. Wu, T. A. Liu, W.-T. Hsu, J. H.-C. Ho, C.-C. Lee, *et al.*, "Performance and limitation of machine learning algorithms for diabetic retinopathy screening: meta-analysis," *Journal of medical Internet research*, vol. 23, no. 7, p. e23863, 2021.

[4] X. Zheng, C. Zhang, and P. C. Woodland, "Adapting gpt, gpt-2 and bert language models for speech recognition," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 162–168, IEEE, 2021.

[5] M. Naseer, M. Asvial, and R. F. Sari, "An empirical comparison of bert, roberta, and electra for fact verification," in *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 241–246, IEEE, 2021.

[6] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, Association for Computational Linguistics, Oct. 2014.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] V. Mnih, N. Heess, A. Graves, *et al.*, "Recurrent models of visual attention," *Advances in neural information processing systems*, vol. 27, 2014.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[11] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *International Conference on Machine Learning*, pp. 10524–10533, PMLR, 2020.

[12] A. Baevski and M. Auli, "Adaptive input representations for neural language modeling," *arXiv preprint arXiv:1809.10853*, 2018.

[13] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[14] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao, "Learning deep transformer models for machine translation," *arXiv preprint arXiv:1906.01787*, 2019.

[15] D. Hendrycks and K. Gimpel, "Bridging nonlinearities and stochastic regularizers with gaussian error linear units," *CoRR*, vol. abs/1606.08415, 2016.

[16] D. So, W. Mańke, H. Liu, Z. Dai, N. Shazeer, and Q. V. Le, "Searching for efficient transformers for language modeling," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6010–6022, 2021.

[17] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[18] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[19] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li, "On the sentence embeddings from pre-trained language models," *arXiv preprint arXiv:2011.05864*, 2020.

[20] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

Everyone can NLP A User-defined Well Visualized NLP Training Platform

ORIGINALITY REPORT

14%	7%	6%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Hong Kong Student Paper	7%
2	www.arxiv-vanity.com Internet Source	<1%
3	Yunhui Zeng, Zijun Liao, Xiu Li, Bo Yuan. "You Only Train Once: A highly generalizable reinforcement learning method for dynamic job shop scheduling problem", Institute of Electrical and Electronics Engineers (IEEE), 2022 Publication	<1%
4	assets.amazon.science Internet Source	<1%
5	papers.neurips.cc Internet Source	<1%
6	Submitted to University of Newcastle upon Tyne Student Paper	<1%
7	curve.carleton.ca	

	Internet Source	<1%
8	norma.ncirl.ie Internet Source	<1%
9	cs.adelaide.edu.au Internet Source	<1%
10	Chujie Zheng, Kunpeng Zhang, Harry Jiannan Wang, Ling Fan, Zhe Wang. "Enhanced Seq2Seq Autoencoder via Contrastive Learning for Abstractive Text Summarization", 2021 IEEE International Conference on Big Data (Big Data), 2021 Publication	<1%
11	arxiv.org Internet Source	<1%
12	"ECAI 2020", IOS Press, 2020 Publication	<1%
13	Minds.wisconsin.edu Internet Source	<1%
14	Arij Al Adel, Mikhail S. Burtsev. "Memory transformer with hierarchical attention for long document processing", 2021 International Conference Engineering and Telecommunication (En&T), 2021 Publication	<1%
	github.com	

15	Internet Source	<1%
16	Submitted to University of Nottingham Student Paper	<1%
17	spectrum.library.concordia.ca Internet Source	<1%
18	David Patterson, Joseph Gonzalez, Urs Holzle, Quoc Le et al. "The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink", Computer, 2022 Publication	<1%
19	Hailong Li, Jaewan Choi, Sunjung Lee, Jung Ho Ahn. "Comparing BERT and XLNet from the Perspective of Computational Characteristics", 2020 International Conference on Electronics, Information, and Communication (ICEIC), 2020 Publication	<1%
20	Submitted to Segi University College Student Paper	<1%
21	yourflorenceguide.com Internet Source	<1%
22	aethos.readthedocs.io Internet Source	<1%
23	mafiadoc.com Internet Source	<1%

24	"Machine Learning for Cyber Security", Springer Science and Business Media LLC, 2020 Publication	<1 %
25	Www.hindawi.com Internet Source	<1 %
26	Submitted to Saimaan ammattikorkeakoulu Student Paper	<1 %
27	pergamos.lib.uoa.gr Internet Source	<1 %
28	Submitted to University of Sheffield Student Paper	<1 %
29	Dengwen Lin, Jintao Tang, Xinyi Li, Kunyuan Pang, Shasha Li, Ting Wang. "BERT-SMAP: Paying attention to Essential Terms in passage ranking beyond BERT", Information Processing & Management, 2022 Publication	<1 %
30	Submitted to QA Learning Student Paper	<1 %
31	Xizhe Wang, Xiaoyong Mei, Qionghao Huang, Zhongmei Han, Changqin Huang. "Fine-grained learning performance prediction via adaptive sparse self-attention networks", Information Sciences, 2021 Publication	<1 %

32	kth.diva-portal.org Internet Source	<1 %
33	Submitted to 6908 Student Paper	<1 %
34	core.ac.uk Internet Source	<1 %
35	paperswithcode.com Internet Source	<1 %
36	Ilenia Fronza, Andrea Janes, Alberto Sillitti, Giancarlo Succi, Stefano Trebeschi. "Cooperation wordle using pre-attentive processing techniques", 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2013 Publication	<1 %
37	docs.google.com Internet Source	<1 %
38	upcommons.upc.edu Internet Source	<1 %
39	web.archive.org Internet Source	<1 %
40	huggingface.co Internet Source	<1 %

Exclude quotes	Off	Exclude matches	< 4 words
Exclude bibliography	Off		