

Large Language Models Open New Way of AI-Assisted Molecule Design for Chemists

Shoichi Ishida,^{*,†,‡} Tomohiro Sato,[¶] Teruki Honma,[¶] and Kei Terayama^{*,†,§,||,‡}

[†]*Graduate School of Medical Life Science, Yokohama City University, 1-7-29, Suehiro-cho, Tsurumi-ku, Yokohama 230-0045 Kanagawa, Japan*

[‡]*MolNavi LLC, #402 Wizard building 1-4-3 Sengen-cho Nishi-ku, Yokohama 220-0072 Kanagawa, Japan*

[¶]*RIKEN Center for Life Science Technologies, 1-7-22 Suehiro-cho, Tsurumi-ku, Yokohama 230-0045, Japan*

[§]*RIKEN Center for Advanced Intelligence Project, 1-4-1, Nihonbashi, Chuo-ku, 103-0027 Tokyo, Japan*

^{||}*MDX Research Center for Element Strategy, Tokyo Institute of Technology, 4259, Nagatsuta-cho, Midori-ku, Yokohama, 226-8501 Kanagawa, Japan*

E-mail: ishida.sho.nm@yokohama-cu.ac.jp; terayama@yokohama-cu.ac.jp

Abstract

Recent advancements in artificial intelligence (AI)-based molecular design methodologies have offered synthetic chemists new ways to design functional molecules with their desired properties. While various AI-based molecule generators have significantly advanced toward practical applications, their effective use still requires specialized knowledge and skills concerning AI techniques. Here, we develop a large language model (LLM)-powered chatbot, ChatChemTS, that enables chemists to design new molecules using an AI-based molecule generator through only chat interac-

tions, including automated construction of reward functions for the specified properties. Our study showcases the utility of ChatChemTS through de novo design cases involving chromophores and anticancer drugs (epidermal growth factor receptor inhibitors), exemplifying single- and multiobjective molecule optimization scenarios, respectively. ChatChemTS is provided as an open-source package on GitHub at <https://github.com/molecule-generator-collection/ChatChemTS>.

Introduction

Artificial intelligence (AI)-based techniques for molecular designs are becoming promising methods for designing synthetically accessible and insightful molecules with desired functionalities.¹⁻⁸ Research articles on these techniques have been reported in a wide range of fields, from material design to drug discovery. In terms of material design, fluorescent⁴ and photofunctional^{1,5} molecules have been designed using AI-based molecule generators, and the designed molecules were successfully experimentally validated to exhibit the desired properties. Similarly, in drug discovery, new proton pump inhibitors⁶ and inhibitors for targeting antifibrotic effects⁷ were designed and demonstrated their good inhibitory effects. The AI-based molecule generators used in the above studies represent just a fraction of the techniques that have been developed thus far,⁹⁻²⁵ and applying and testing various promising molecule generators to solve real-world problems is vital for achieving further advancements.

While various AI-based molecule generators have made significant progress toward practical applications, their effective utilization still requires specialized knowledge and skills concerning AI techniques.²⁶ This high level of expertise presents a critical obstacle to the widespread adoption of AI-based molecule generators. The effective use of these methods necessitates a deep understanding of how to design reward functions that appropriately represent the desired functionalities and the ability to configure the set conditions according to the specifications of each AI-based molecule generator. In chemical, pharmaceutical, and other industries, the complexity of utilizing AI-based molecule generators and the need for

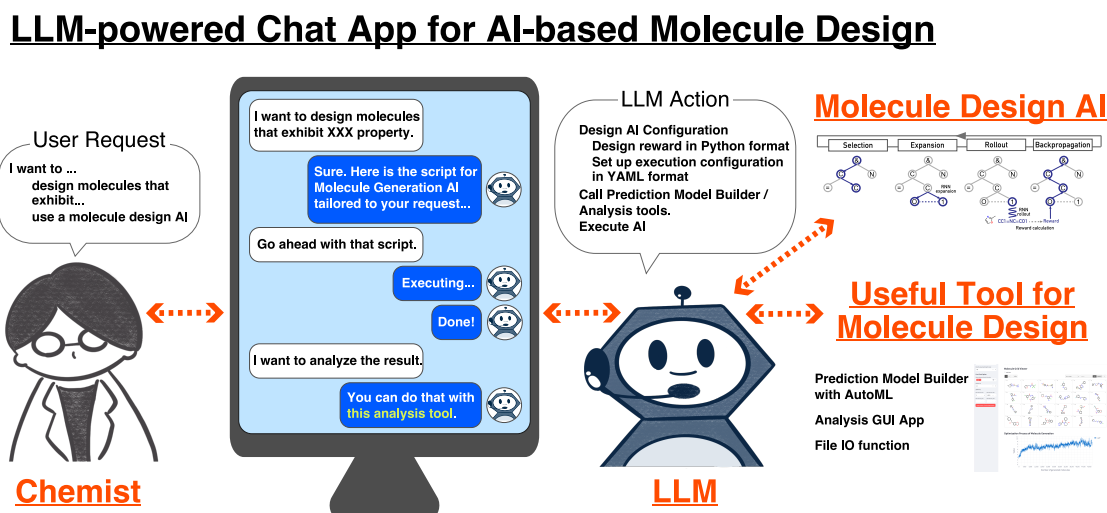
skills such as machine learning (ML,) to design reward functions pose significant obstacles that prevent users from easily adopting these technologies for their projects. These challenges complicate the effective utilization of AI-based molecule generators to solve real-world problems, especially for researchers and developers who possess expert knowledge and skills in chemistry but are not well versed in AI techniques.

To address these challenges, we developed ChatChemTS, a large language model (LLM)-powered chatbot that enables users to utilize ChemTSv2¹¹—AI-based molecule generator with experimental validations for various molecule designs^{1,3-5}—through only interactive chats. All users are merely required to express a request to ChatChemTS via chat, and ChatChemTS then prepares the appropriate reward functions, configures the desired conditions, and executes ChemTSv2 for the users. In addition, ChatChemTS provides a tool for analyzing the output molecule generation results. ChatChemTS is based on a ReAct framework so that it can address the whole workflow of general AI-based molecule generators, and the framework employs the generative pretrained transformer (GPT) model of OpenAI, which has shown the great potential as an LLM chemistry agent to perform chemistry-related tasks.²⁷⁻³³ As example applications of ChatChemTS, we performed two de novo molecular design tasks, one involving a photofunctional organic molecule and another concerning a kinase inhibitor, as single- and multiobjective molecule optimization problems, respectively. Notably, users only need to prepare data related to the physicochemical properties of molecules or information about the target proteins of interest to perform AI-based molecule designs with ChatChemTS. We show that this concept of utilizing an LLM as an assistant of AI-based molecule generators can be easily introduced to various generators developed with organized application structures, such as ChemTSv2. The ChatChemTS application is publicly available on GitHub at <https://github.com/molecule-generator-collection/ChatChemTS>.

Results

Implementation of ChatChemTS

ChatChemTS was developed based on LLMs to help users employ ChemTSv2 through interactive chats, as shown in Fig. 1. The ChatChemTS application employs a ReAct framework³⁴



Application UI Snapshot

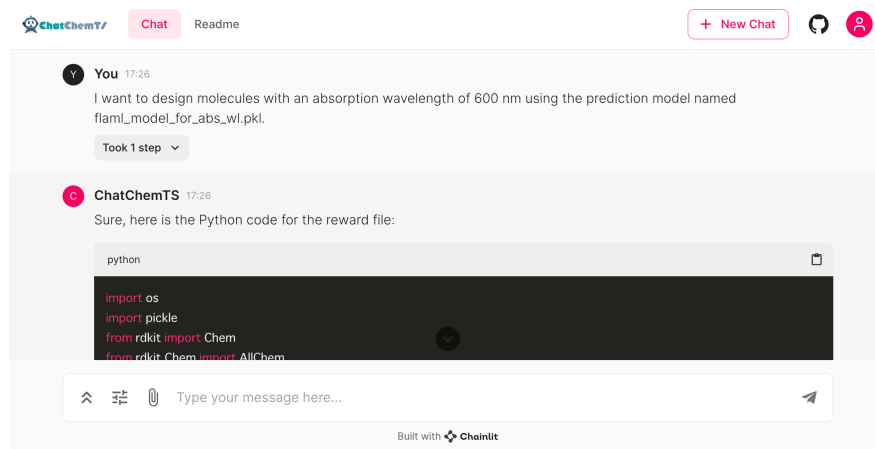


Figure 1: Overview of ChatChemTS. The visual workflow of ChatChemTS is shown in the upper panel. A user can utilize ChatChemTS via chat in a web browser on a local laptop, and ChatChemTS assists users in designing molecules through AI. A snapshot of the user interface (UI) of this application is shown in the lower panel. The UI was built with Chainlit and provides the users with an intuitive chat experience.

that enables LLMs to generate reasonable responses and take appropriate actions, including

the use of predefined tools, as shown in Fig. 2. The predefined tools in ChatChemTS include a reward generator, a prediction model builder, a configuration generator, a ChemTSv2 application programming interface (API), a molecule generation analyzer, and a file writing tool. In this study, the configuration included molecule generation parameters in ChemTSv2. To offer high-quality responses, multiple LLMs were utilized in ChatChemTS and specifically tailored for distinct roles, such as facilitating user interactions and crafting reward function designs; thus, the use of verbose and ambiguous prompts that may elicit irrelevant responses was minimized in each LLM.

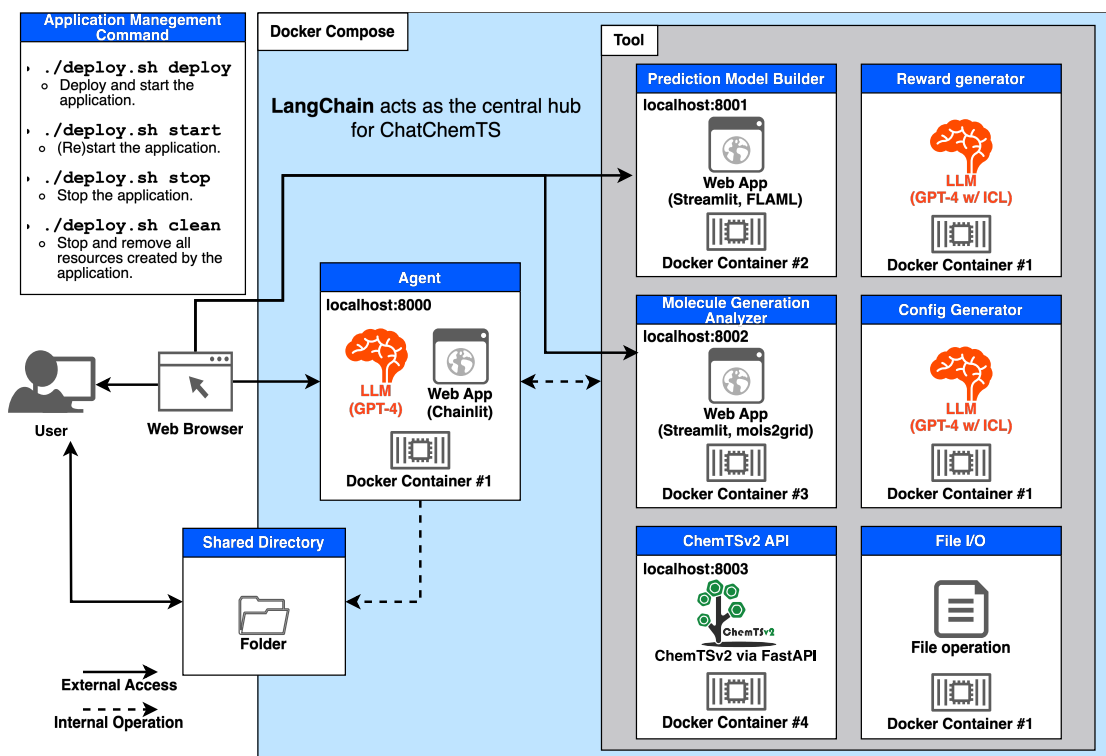


Figure 2: System architecture of ChatChemTS. LangChain was used to implement the ReAct framework in ChatChemTS. LLMs were used in the agent, the reward generator tool, and the configuration generator tool. Additionally, both tools utilized LLMs specialized in the corresponding tasks using in-context learning (ICL). ChatChemTS was containerized using Docker, allowing for easy deployment to user environments through Docker Compose. Users can employ ChatChemTS via local web browsers, such as Google Chrome.

The platform was developed primarily by incorporating LangChain,³⁵ GPT-4 from OpenAI,³⁶ and FastAPI³⁷ for backend operations, as well as Streamlit³⁸ and Chainlit³⁹ for

frontend implementations, as shown in Fig. 2. To enhance the scalability of the application, the computing environments assigned to each role were containerized using Docker, ensuring that no interference occurred among them. Docker Compose⁴⁰ was utilized to implement a streamlined deployment process leveraging multiple Docker images, facilitating the automatic deployment of the application in a Docker-compatible environment. A demonstration of the workflow for ChatChemTS can be found in Supporting Movie 1.

Exemple applications of ChatChemTS

To demonstrate the utility of ChatChemTS, we performed de novo designs of a chromophore and an epidermal growth factor receptor (EGFR) inhibitor, the latter of which is a common treatment for breast and lung cancers.⁴¹ These designs illustrated examples of single- and multiobjective molecule optimization tasks, respectively. The aim of the chromophore design process was to optimize molecules so that they possessed a specific absorption wavelength. On the other hand, the EGFR inhibitor design process focused on obtaining molecules with high inhibitory activity and high drug likeness scores. Detailed explanations of both demonstrations are described below.

Chromophore design

Figure 3 shows a chat-based demonstration of the task of designing chromophores using ChatChemTS. All processes described hereafter were successfully carried out through chat interactions and operations in graphical user interface (GUI) applications: prediction model builder and analysis tools. The prediction model builder of ChatChemTS provides a function for constructing ML models that predict a molecule property when provided a dataset in comma-separated values (CSV) format (see the Methods section for details). First, the prediction model builder tool was used to create an ML model for predicting an absorption wavelength from an input molecule. The training dataset was prepared in CSV format and included 50000 molecules with absorption wavelengths calculated using density functional

STEP 1. Reward Design Task

I want to create a prediction model using my own dataset.

Open FLAML Model Builder in your browser to create prediction models.

[Prediction Model Builder Tool]
Obtain the model to predict an absorption wavelength from an input molecule.

I want to design molecules with an absorption wavelength of 600 nm using the prediction model named flaml_model_for_abs_wl.pkl.

```
python
import os
import pickle
from rdkit import Chem
from rdkit.Chem import AllChem
import numpy as np
from chemtsv2.reward import Reward

FLAML_MODEL_NAME = 'flaml_model_for_abs_wl.pkl' #
User specified filename.
with open(os.path.join('/app/shared_dir',
FLAML_MODEL_NAME), 'rb') as f:
    AUTOML = pickle.load(f)

class CustomReward(Reward):
    def get_objective_functions(conf):
        def AbsorptionWavelength(mol):
            fp = np.array(
                (AllChem.GetMorganFingerprintsAsBitVect(mol, 2,
                2848))[np.newaxis, :])
            return AUTOML.predict(fp).item()
            return AbsorptionWavelength

        def calc_reward_from_objective_values(values,
        conf):
            def gaussian(x, mu):
                sigma = mu * 0.2
                return np.exp(-(x - mu) ** 2) / (2 *
                sigma ** 2)
            return gaussian(values[0], 600)
            ...

In this script, the AbsorptionWavelength function calculates the absorption wavelength of a molecule using the FLAML model. The calc_reward_from_objective_values function then calculates the reward for a molecule based on how close its absorption wavelength is to 600 nm, using a Gaussian function. The closer the absorption wavelength is to 600 nm, the higher the reward.
```

Save the above reward script as reward_abs_wl_600.py.

The file has been successfully saved at shared_dir/reward_abs_wl_600.py.

STEP 2. Config Setup Task

Write a config file to generate 30000 molecules using the reward file, reward_abs_wl_600.py with SAScore filter set to 4.5. Save the output in output_abs_wl_600.

```
yaml
c_val: 0.1
# threshold types: [time, generation_num]
threshold type: generation_num
# hours: 0.01
generation_num: 30000
output_dir: shared_dir/output_abs_wl_600
model_setting:
  model_json: model/model.tf25.json
  model_weight: model/model.tf25.bestckpt.h5
  token: model/tokens.pkl
reward_setting:
  reward_module: shared_dir/reward_abs_wl_600
  reward_class: CustomReward
  use_lipinski_filter: False
lipinski_filter:
  module: filter.lipinski_filter
  class: LipinskiFilter
  type: rule_of_5
  use_radical_filter: False
radical_filter:
  module: filter.radical_filter
  class: RadicalFilter
  use_pubchem_filter: False
pubchem_filter:
  module: filter.pubchem_filter
  class: PubChemFilter
  use_sascore_filter: True
sascore_filter:
  module: filter.sascore_filter
  class: SAScoreFilter
  threshold: 4.5

In this configuration file, generation_num is set to 30000, which means the generator will create 30000 molecules. The output_dir is set to shared_dir/output_abs_wl_600, so the output will be saved in this directory.

The reward_module is set to shared_dir/reward_abs_wl_600, which means the generator will use the reward function defined in the reward_abs_wl_600.py file in the shared_dir directory.

The use_sascore_filter is set to True, and the threshold under sascore_filter is set to 4.5, which means the generator will use the SAScore filter with a threshold of 4.5. The other filters (Lipinski, Radical, and PubChem) are set to False, which means they will not be used.
```

Save the above config file as config_abs_wl_600.yaml.

The file has been successfully saved at shared_dir/config_abs_wl_600.yaml.

STEP 3. Execution Task

Run ChemTSv2 using config_abs_wl_600.yaml.

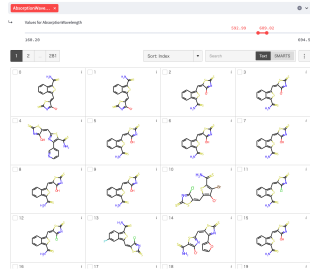
The ChemTSv2 application has been successfully run. The output results are stored in the directory shared_dir/output_abs_wl_600.

STEP 4. Result Analysis Task

I want to analyze the result.

Open ChatChemTS Analysis App in your browser to analyze the molecule generation result.

Molecule Grid Viewer



Optimization Process of Molecule Generation

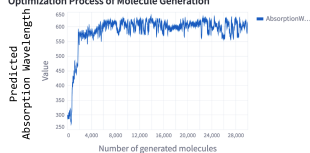


Figure 3: Application of ChatChemTS in designing chromophore. This demonstration aimed to design molecules with absorption wavelengths of 600 nm. In step 1, based on the observed user requests, ChatChemTS created a prediction model to predict the absorption wavelength from an input molecule. The training dataset was prepared using 50000 compounds with DFT-calculated absorption wavelengths at the B3LYP/6-31G* level⁴² via the prediction model builder. The prediction model was used in the reward function. In step 2, ChatChemTS set up the configuration of ChemTSv2 according to the specifications provided by the user. In step 3, ChatChemTS executed the ChemTSv2 using the prepared configuration file. In step 4, a user analyzed the molecule generation results. The right panel shows examples of molecules with absorption wavelengths of approximately 600 nm and the optimization process of the molecule generation task.

theory (DFT) at the B3LYP/6-31G* level.⁴² The AutoML parameters were set with a test dataset ratio of 0.1 and a budget time of one hour; the estimators list and metrics were automatically selected by the Fast Library for Automated Machine Learning and Tuning (FLAML) during the AutoML search process. The best model was the light gradient boosting machine (LightGBM), and its correlation coefficient for the test dataset was 0.93. The above operations and their results are shown in Fig. S1. Next, a reward function and a configuration were designed via chat based on the following conditions: the target absorption wavelength was set to 600 nm, the exploration parameter c was set to 0.1, the number of generated molecules was set to 30000, and an SAScore filter with a threshold of 4.5 was used. Then, ChemTSv2 was executed via chat using the above reward function and configuration files. Finally, the analysis tool was utilized to analyze the molecule generation results. As shown in the optimization process of Fig. 3 (right panel), ChatChemTS successfully designed molecules with predicted absorption wavelength of approximately 600 nm. These seemingly complex processes for chemists can now be effortlessly automated through ChatChemTS, simply by inputting the dataset and following the procedure outlined in the GUI.

EGFR inhibitor design

Figure 4 shows a chat-based demonstration of the task of designing EGFR inhibitors using ChatChemTS. The subsequent processes were all accomplished through chat interactions and operations in the GUI applications. The prediction model builder of ChatChemTS provides a function for constructing ML models that predict inhibitory activity against a target protein by simply specifying its Universal Protein Resource ID (UniProt ID; see the Methods section for details). In this demonstration, UniProt ID P00533 was used as the input of the prediction model builder tool to retrieve molecules with inhibitory activities against EGFR from the ChEMBL database. ML models were designed to take a molecule as input and predict a pChEMBL value, representing the negative base-10 logarithm of the half-maximal response concentration, potency, and affinity. The retrieved dataset was preprocessed as follows: (1)

STEP 1. Reward Design Task

I want to create a prediction model using my own dataset.

Open FLAML Model Builder in your browser to create prediction models.

(Prediction Model Builder Tool)
Obtain the model to predict an inhibitory activity against EGFR from an input molecule.

I want to design molecules with high inhibitory activity against EGFR and a high QED score. Use the prediction model named flaml_model_egfr_qsar.pkl to estimate the inhibitory activity.

Sure, here is a Python script that defines a custom reward function using the prediction model flaml_model_egfr_qsar.pkl to maximize inhibitory activity against EGFR and the QED score.

```
... python
import os
import pickle
from rdkit import Chem
from rdkit.Chem import AllChem, QED
import numpy as np
from chemsv2.reward import Reward

FLAML_MODEL_NAME = 'flaml_model_egfr_qsar.pkl'
with open(os.path.join('/app/shared_dir',
FLAML_MODEL_NAME), 'rb') as f:
    AUTOML = pickle.load(f)

class CustomReward(Reward):
    def get_objective_functions(conf):
        def PredictedEGFRInhibition(mol):
            fp = np.array
            [AllChem.GetMorganFingerprintsAsBitVect(mol, 2,
2048)][np.newaxis, :]
            return AUTOML.predict(fp).item()

        def QEDScore(mol):
            return QED.qed(mol)

        return [PredictedEGFRInhibition, QEDScore]

    def calc_reward_from_objective_values(values,
conf):
        return np.mean([np.tanh(value) for value in
values])

In this script, get_objective_functions() returns two
functions: PredictedEGFRInhibition() and QEDScore(). The
former uses the FLAML model to predict the inhibitory
activity against EGFR, and the latter calculates the QED
score of the molecule.

In calc_reward_from_objective_values(), each objective
value is preprocessed using the hyperbolic tangent
function to scale it to the range of -1 to 1. The final reward
is the average of these preprocessed values. This ensures
that the reward is a balance between the two objectives:
maximizing EGFR inhibitory activity and maximizing the
QED score.
```

Save the above reward script as reward_egfr_qsar_and_qed.py.

The file has been successfully saved at shared_dir/reward_egfr_qsar_and_qed.py.

STEP 2. Config Setup Task

Write a config file to generate 50000 molecules using the reward file, reward_egfr_qsar_and_qed.py with SAScore filter set to 5.0 and Lipinski's rule of five filter. Save the output in output_egfr_qsar_and_qed.

Sure, here is the configuration file based on your specifications:

```
... yaml
c_val: 0.1
threshold_type: generation_num
generation_num: 50000
output_dir: shared_dir/output_egfr_qsar_and_qed
model_setting:
  model_json: model/model.tf25.json
  model_weight: model/model.tf25.best.ckpt.h5
  token: model/tokens.pkl
reward_setting:
  reward_module:
    shared_dir.reward_egfr_qsar_and_qed
  reward_class: CustomReward
  use_lipinski_filter: True
  lipinski_filter:
    module: filter.lipinski_filter
    class: LipinskiFilter
    type: rule of 5
  use_radical_filter: False
  radical_filter:
    module: filter.radical_filter
    class: RadicalFilter
  use_pubchem_filter: False
  pubchem_filter:
    module: filter.pubchem_filter
    class: PubChemFilter
  use_sascore_filter: True
  sascore_filter:
    module: filter.sascore_filter
    class: SascoreFilter
    threshold: 5.0
... 
```

This configuration file will generate 50000 molecules using the reward file reward_egfr_qsar_and_qed.py. The SAScore filter is set to 5.0 and the Lipinski's rule of five filter is enabled. The output will be saved in the directory shared_dir/output_egfr_qsar_and_qed. Please note that the reward_module is set to shared_dir.reward_egfr_qsar_and_qed as per your request. This assumes that the reward_egfr_qsar_and_qed.py file is located in the shared_dir directory.

Save the above config file as config_egfr_qsar_and_qed.yaml.

The file has been successfully saved at shared_dir/config_egfr_qsar_and_qed.yaml.

STEP 3. Execution Task

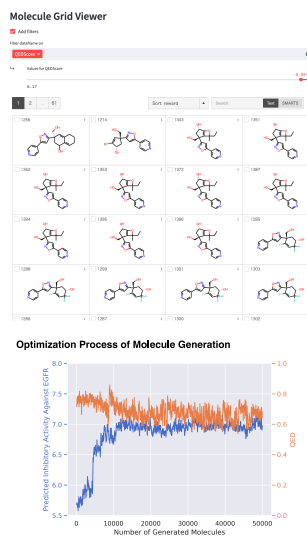
Run ChemTSv2 using config_egfr_qsar_and_qed.yaml.

The ChemTSv2 application has been successfully run. The output results are stored in the directory shared_dir/output_egfr_qsar_and_qed.yaml.

I want to analyze the result.

Open ChatChemTS Analysis App in your browser to analyze the molecule generation result.

STEP 4. Result Analysis Task



deduplicating molecules by leaving the maximum pChEMBL values; (2) retaining records by assay type of Binding; (3) filtering out records with assay descriptions, which contained the mutat, covalent, and irreversible substrings; and (4) removing the activity types of the half-maximal effective concentration (EC50) and half-maximal active concentration (AC50). Consequently, the dataset size was 7141 compounds, and the dataset was used in the training step. The AutoML parameters were set with a test dataset ratio of 0.1 and a budget time of one hour; the estimator list and metrics were automatically selected by the FLAML during the AutoML search process. The option to standardize the objective variable was applied to facilitate the use of the prediction values within a reward function. The best model was the LightGBM, and its correlation coefficient for the test dataset was 0.85. The above operations and their results are shown in Fig. S2. Next, a reward function and a configuration were designed via chat interactions based on the following conditions: maximizing the predicted inhibitory activity against EGFR and the QED score; setting the exploration parameter c to 0.1; setting the number of generated molecules to 50000; and using Lipinski's rule-of-five filter and the SAScore filter with a threshold of 4.5. Then, ChemTSv2 was executed via chat using the above reward function and configuration files. Finally, the analysis tool was utilized to analyze the molecule generation results. As shown in the optimization process of Fig. 3 (right panel), ChatChemTS successfully designed molecules with predicted pChEMBL values above seven and QED scores of approximately 0.7. To confirm whether the molecule generation process considering the QED score worked properly, we compared the QED optimization processes of methods generating molecules based on both inhibitory activity and QED scores and methods based only on inhibitory activity under the same conditions. Figure S3 indicates that the molecules designed solely based on inhibitory activity often had QED scores of approximately 0.4, highlighting the effectiveness of incorporating QED scores with inhibitory activity in the reward function.

Discussion and Conclusion

In this study, we introduced ChatChemTS, an LLM-powered application, to facilitate user interactions with ChemTSv2 through an interactive chat interface. ChatChemTS successfully assisted with four main operations that users need to perform when using AI-based molecule generators: designing reward functions, setting up configurations, executing a molecule generator, and analyzing the results. Two demonstrations, chromophore and EGFR inhibitor design tasks, were showcased as common de novo molecular generation tasks: single- and multiobjective optimizations, respectively. Despite their success in terms of designing molecules with targeted properties based on prediction models, there is potential for improving the design of the reward functions. For example, the designed EGFR inhibitor lacked the major hinge-binding scaffolds used in common kinase inhibitors.⁴³ A potential solution to this issue is to introduce a reward function designed to increase the structural similarity to these scaffolds, which is an unsupported feature in ChatChemTS.

The following tasks remain to make the experience of AI-based molecule designs via ChatChemTS more appropriate and convenient. ChatChemTS currently supports a single AI-based molecule generator, ChemTSv2; however, ChatChemTS was designed to easily incorporate additional molecule generators, assuming that the reward design and configuration settings are independently specified in separate files. With respect to the topic of processor compatibility, ChatChemTS supports processors with x86_64 architectures but lacks support for the arm64 architecture, which is attributed to the difficulty of configuring Python environments. Regarding users' experience during chat interactions, ChatChemTS retains only the most recent chat interaction to ensure the relevance of its responses. This limitation may lead to unexpected behaviors for users who are unaware of this specification. In terms of reward designs, the current reward generator tool is limited to using Python packages and ML-based prediction models within reward functions. Given the frequent use of various simulation packages, such as the Gaussian 16⁴⁴ and AutoDock Vina⁴⁵ packages, in material and drug design scenarios, we plan to make these packages available in the reward designs

of ChatChemTS. Furthermore, introducing a feature that automatically optimizes reward designs⁴⁶ could significantly reduce the manual effort required for reward adjustment, as this process typically involves extensive trial and error. In the configuration setup stage, switching the recurrent neural network (RNN) models provided in ChemTSv2 via ChatChemTS is difficult unless users are familiar with ChemTSv2. Efforts are underway to simplify the method for specifying the desired RNN model to address this issue. While ChatChemTS currently supports the basic use cases of ChemTSv2 and still requires expert intervention for comprehensive AI-based molecule designs, it is intended to lay the groundwork for LLM-assisted molecule designs utilizing various AI-based molecule generators.

Methods

Large language model

An LLM is a type of AI model that can perform various general-purpose natural language processing (NLP) tasks at the human level, including text generation, question answering, and information extraction.⁴⁷ The core architecture of an LLM is a deep learning technique called a transformer,⁴⁸ and LLMs are generally trained on immense amounts of data. LLMs are frequently utilized in combination with prompting strategies, such as ICL, Chain of Thought (CoT), and Planning, for solving various tasks to enhance their contextual understanding and improve their task-specific performance.⁴⁷ In this study, GPT-4 developed by OpenAI with a temperature of 0.1 and the ReAct framework were used as the LLM and the Planning strategy, respectively.

ReAct framework

The ReAct framework allows LLMs to intertwine reasoning traces with task-specific actions in external environments, facilitating general task-solving.³⁴ The LLM that performs the above role is called an agent. Within the ReAct framework, LLMs can interact with external

tools, such as Wikipedia, web searches, the Python interactive computing environment, and user-customized tools, and use their feedback for reasoning traces to produce more reliable responses. LangChain was used to implement the ReAct framework in ChatChemTS. The system message used in the agent is shown in Listing S1. The following section describes the tools used in ChatChemTS.

Tools

In the ReAct framework, tools are pivotal for enhancing LLMs by enabling the retrieval of additional information, which contributes to more reliable responses. Notably, the tools included in the proposed application were the minimum tools required for using AI-based molecule generators, but other tools can easily be added to the application to satisfy users' specific needs.

Reward generator

This tool is dedicated to designing reward functions in ChemTSv2 format and is based on an LLM. A few-shot prompting technique was used to steer the LLM to the reward design tool. All the utilized prompts are described in Listing S2. Upon receiving a user request for molecules optimized based on specific properties, this tool returns a reward function reflecting that request. The currently available molecular properties are those that can be calculated by RDKit software and predicted using prediction models obtained from the FLAML.⁴⁹

Prediction model builder

This tool offers a GUI application that allows users to build their own prediction models using an automated ML tool (the FLAML).⁴⁹ The interface was designed to be accessible to users with varying levels of ML expertise. The application accepts two input types: a CSV file for building general prediction models and a UniProt⁵⁰ ID for constructing quantitative structure-activity relationship (QSAR) models from the ChEMBL database.

When a user uploads a CSV file, it is rendered as a table on the interface to enable the user to verify its content. To build prediction models, the user must select the names of two columns: one containing molecule structures in the SMILES format and another containing a target variable. The interface guides the user through a structured configuration process to develop a prediction model with the FLAML. Initially, the user adjusts the data proportion for the test dataset. Following this, the user has the option to select default ML estimators or manually choose from a list option, including a random forest, LightGBM, eXtreme Gradient Boosting (XGBoost), Categorical Boosting (CatBoost), Extremely Randomized Trees (Extra Trees), Logistic Regression with L1 or L2 regularization, and k-nearest neighbors. Then, the user specifies the type of ML task to perform, which is currently limited to regression, and the metric used to evaluate the performance of the constructed model during training, with the option to utilize automatic metric selection. A time budget can be set to manage the amount of time that computational resources are dedicated to performing the AutoML search process. If the values of the target variable are not normalized, the user should employ the standardization function in this application to standardize the target values, ensuring that reward functions can be appropriately designed. Once all the settings are finalized, this tool runs the AutoML process to find quality models and saves the model that performs best on the test dataset for use in a reward function.

When a user inputs a UniProt ID, the application fetches and processes data from the ChEMBL database using the ChEMBL webresource client package.⁵¹ The application verifies the existence of records for the specified UniProt ID and then allows users to refine the dataset by deduplicating molecules based on pChEMBL values, retaining records according to their specific assay types, and filtering out records with certain assay descriptions and activity types. Subsequently, the workflow is the same as that of the process described above after uploading a CSV file.

Configuration generator

This tool is specialized in generating configuration files in YAML format for ChemTSv2 and is based on an LLM. The configuration includes, for example, the number of generated molecules, the exploration parameter c , and molecular filters for skipping the reward calculation (refer to the paper that presented ChemTSv2 for details¹¹). Similar to the reward generator tool, this tool employs a few-shot prompting technique to generate configurations. All the prompts used are described in Listing S3. By taking a user request for setting a ChemTSv2 configuration, this tool returns a configuration reflecting that request.

ChemTSv2 API

This tool serves as an API for executing ChemTSv2. Upon receiving a path to a configuration file, the tool runs a ChemTSv2 using the input configuration and returns a path to the execution outcome. The API was built with the FastAPI package.³⁷

Molecule generation analyzer

This tool provides users with a GUI application to easily analyze the results of molecule generation processes. Once a user uploads a CSV result file, the application offers three interactive functions, a table viewer, a molecule viewer, and a time series viewer, which are the common features used to analyze molecule generation results. The GUI application was developed based on Streamlit³⁸ for creating the user interface, pandas⁵² for manipulating molecule generation results, and mols2grid⁵³ for interactively visualizing molecules.

Data and Software Availability

The ChatChemTS application is publicly available on GitHub at <https://github.com/molecule-generator-collection/ChatChemTS> under the MIT License. The README file in the GitHub repository provides information about how to set up and use the application.

Author Contributions

Shoichi Ishida: conceptualization (lead); methodology (lead); software (lead); writing - original draft (lead); writing - review & editing (lead). **Tomohiro Sato:** methodology (support); software (support), writing -review & editing (support). **Teruki Honma:** supervision (lead); funding acquisition (lead); project administration (lead); software (support); writing -review & editing (supporting). **Kei Terayama:** supervision (lead); funding acquisition (lead); methodology (equal); project administration (lead); conceptualization (lead); software (support); writing - review & editing (lead).

Acknowledgement

This work was conducted in “Development of a Next-generation Drug Discovery AI through Industry-Academia Collaboration (DAIIA)” from Japan Agency for Medical Research and Development (AMED) under grant no.JP23nk0101111 and a Research Support Project for Life Science and Drug Discovery under grant no.JP22ama121023. This work was also supported by MEXT as a “Program for Promoting Researches on the Supercomputer Fugaku (Simulation and AI-driven next-generation medicine and drug discovery based on Fugaku)” (JPMXP1020230120) and Data Creation and Utilization Type Material Research and Development Project Grant Number JPMXP1122683430.

Supporting Information Available

Snapshot UI of prediction model builder tool when building the prediction models to predict absorption wavelength (Fig. S1); Snapshot UI of prediction model builder tool when building the prediction model to predict inhibitory activity against EGFR (Fig. S2); Comparizon of QED optimization processes between generating molecules considering inhibitory activity against EGFR and QED score and generating molecules solely considering the inhibitory

activity (Fig. S3); System message used in the agent of ChatChemTS (Listing S1); Few-shot prompting for the reward generator tool (Listing S2); Few-shot prompting for config generator tool (Listing S3).

References

- (1) Sumita, M.; Yang, X.; Ishihara, S.; Tamura, R.; Tsuda, K. Hunting for Organic Molecules with Artificial Intelligence: Molecules Optimized for Desired Excitation Energies. *ACS Central Science* **2018**, *4*, 1126–1133.
- (2) Kajita, S.; Kinjo, T.; Nishi, T. Autonomous molecular design by Monte-Carlo tree search and rapid evaluations using molecular dynamics simulations. *Communications Physics* **2020**, *3*.
- (3) Zhang, Y.; Zhang, J.; Suzuki, K.; Sumita, M.; Terayama, K.; Li, J.; Mao, Z.; Tsuda, K.; Suzuki, Y. Discovery of polymer electret material via de novo molecule generation and functional group enrichment analysis. *Applied Physics Letters* **2021**, *118*.
- (4) Sumita, M.; Terayama, K.; Suzuki, N.; Ishihara, S.; Tamura, R.; Chahal, M. K.; Payne, D. T.; Yoshizoe, K.; Tsuda, K. De novo creation of a naked eye-detectable fluorescent molecule based on quantum chemical computation and machine learning. *Science Advances* **2022**, *8*.
- (5) Fujita, T.; Terayama, K.; Sumita, M.; Tamura, R.; Nakamura, Y.; Naito, M.; Tsuda, K. Understanding the evolution of a de novo molecule generator via characteristic functional group monitoring. *Science and Technology of Advanced Materials* **2022**, *23*, 352–360.
- (6) Abe, K.; Ozako, M.; Inukai, M.; Matsuyuki, Y.; Kitayama, S.; Kanai, C.; Nagai, C.; Gopalasingam, C. C.; Gerle, C.; Shigematsu, H.; Umekubo, N.; Yokoshima, S.; Yoshi-

- mori, A. Deep learning driven de novo drug design based on gastric proton pump structures. *Communications Biology* **2023**, *6*.
- (7) Ren, F. et al. A small-molecule TNIK inhibitor targets fibrosis in preclinical and clinical models. *Nature Biotechnology* **2024**,
- (8) Ivanenkov, Y.; Zagribelnyy, B.; Malyshev, A.; Evteev, S.; Terentiev, V.; Kamyra, P.; Bezrukov, D.; Aliper, A.; Ren, F.; Zhavoronkov, A. The Hitchhiker's Guide to Deep Learning Driven Generative Chemistry. *ACS Medicinal Chemistry Letters* **2023**, *14*, 901–915.
- (9) Segler, M. H. S.; Kogej, T.; Tyrchan, C.; Waller, M. P. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Central Science* **2017**, *4*, 120–131.
- (10) Yang, X.; Zhang, J.; Yoshizoe, K.; Terayama, K.; Tsuda, K. ChemTS: an efficient python library for de novo molecular generation. *Science and Technology of Advanced Materials* **2017**, *18*, 972–976.
- (11) Ishida, S.; Aasawat, T.; Sumita, M.; Katouda, M.; Yoshizawa, T.; Yoshizoe, K.; Tsuda, K.; Terayama, K. ChemTSv2: Functional molecular design using de novo molecule generator. *WIREs Computational Molecular Science* **2023**, *13*.
- (12) Dollar, O.; Joshi, N.; Beck, D. A. C.; Pfaendtner, J. Attention-based generative models for de novo molecular design. *Chemical Science* **2021**, *12*, 8362–8372.
- (13) Bagal, V.; Aggarwal, R.; Vinod, P. K.; Priyakumar, U. D. MolGPT: Molecular Generation Using a Transformer-Decoder Model. *Journal of Chemical Information and Modeling* **2021**, *62*, 2064–2076.
- (14) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder.

- Proceedings of the 34th International Conference on Machine Learning. 2017; pp 1945–1954.
- (15) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science* **2018**, *4*, 268–276.
- (16) Zang, C.; Wang, F. MoFlow: an invertible flow model for generating molecular graphs. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020; pp 617–626.
- (17) Shi*, C.; Xu*, M.; Zhu, Z.; Zhang, W.; Zhang, M.; Tang, J. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. International Conference on Learning Representations. 2020.
- (18) Guan, J.; Qian, W. W.; Peng, X.; Su, Y.; Peng, J.; Ma, J. 3D Equivariant Diffusion for Target-Aware Molecule Generation and Affinity Prediction. The Eleventh International Conference on Learning Representations. 2023.
- (19) Bao, F.; Zhao, M.; Hao, Z.; Li, P.; Li, C.; Zhu, J. Equivariant Energy-Guided SDE for Inverse Molecular Design. The Eleventh International Conference on Learning Representations. 2023.
- (20) Olivecrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics* **2017**, *9*.
- (21) Loeffler, H. H.; He, J.; Tibo, A.; Janet, J. P.; Voronov, A.; Mervin, L. H.; Engkvist, O. Reinvent 4: Modern AI-driven generative molecule design. *Journal of Cheminformatics* **2024**, *16*.

- (22) Jensen, J. H. A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. *Chemical Science* **2019**, *10*, 3567–3572.
- (23) Nigam, A.; Pollice, R.; Aspuru-Guzik, A. Parallel tempered genetic algorithm guided by deep neural networks for inverse molecular design. *Digital Discovery* **2022**, *1*, 390–404.
- (24) Yoshizawa, T.; Ishida, S.; Sato, T.; Ohta, M.; Honma, T.; Terayama, K. Selective inhibitor design for kinase homologs using multiobjective Monte Carlo tree search. *Journal of Chemical Information and Modeling* **2022**, *62*, 5351–5360.
- (25) Murakami, Y.; Ishida, S.; Demizu, Y.; Terayama, K. Design of antimicrobial peptides containing non-proteinogenic amino acids using multi-objective Bayesian optimisation. *Digital Discovery* **2023**, *2*, 1347–1353.
- (26) Anstine, D. M.; Isayev, O. Generative Models as an Emerging Paradigm in the Chemical Sciences. *Journal of the American Chemical Society* **2023**, *145*, 8736–8750.
- (27) Boiko, D. A.; MacKnight, R.; Kline, B.; Gomes, G. Autonomous chemical research with large language models. *Nature* **2023**, *624*, 570–578.
- (28) Fang, Y.; Liang, X.; Zhang, N.; Liu, K.; Huang, R.; Chen, Z.; Fan, X.; Chen, H. Mol-Instructions - A Large-Scale Biomolecular Instruction Dataset for Large Language Models. The Twelfth International Conference on Learning Representations. 2024.
- (29) Bran, A. M.; Cox, S.; Schilter, O.; Baldassari, C.; White, A.; Schwaller, P. Augmenting large language models with chemistry tools. 37th Conference on Neural Information Processing Systems (NeurIPS 2023) AI for Science Workshop. 2023.
- (30) Song, Y.; Miret, S.; Liu, B. MatSci-NLP: Evaluating Scientific Language Models on Materials Science Language Tasks Using Text-to-Schema Modeling. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2023.

- (31) Guo, T.; Guo, K.; Nan, B.; Liang, Z.; Guo, Z.; Chawla, N. V.; Wiest, O.; Zhang, X. What can Large Language Models do in chemistry? A comprehensive benchmark on eight tasks. Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track. 2023.
- (32) Deb, J.; Saikia, L.; Dihingia, K. D.; Sastry, G. N. ChatGPT in the Material Design: Selected Case Studies to Assess the Potential of ChatGPT. *Journal of Chemical Information and Modeling* **2024**, *64*, 799–811.
- (33) Hasrod, T.; Nuapia, Y. B.; Tutu, H. ChatGPT Helped Me Build a Chemistry App, and Here’s How You Can Make One Also. *Journal of Chemical Education* **2024**, *101*, 653–660.
- (34) Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K. R.; Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. The Eleventh International Conference on Learning Representations. 2023.
- (35) Chase, H. LangChain. 2022; <https://github.com/langchain-ai/langchain>.
- (36) OpenAI GPT-4. 2023; <https://openai.com/gpt-4>.
- (37) Ramírez, S. FastAPI. 2023; <https://github.com/tiangolo/fastapi>.
- (38) Streamlit. 2023; <https://streamlit.io/>.
- (39) Chainlit. 2023; <https://docs.chainlit.io/>.
- (40) Merkel, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.* **2014**, *2014*.
- (41) Zubair, T.; Bandyopadhyay, D. Small Molecule EGFR Inhibitors as Anti-Cancer Agents: Discovery, Mechanisms of Action, and Opportunities. *International Journal of Molecular Sciences* **2023**, *24*, 2651.

- (42) Terayama, K.; Sumita, M.; Tamura, R.; Payne, D. T.; Chahal, M. K.; Ishihara, S.; Tsuda, K. Pushing property limits in materials discovery via boundless objective-free exploration. *Chemical Science* **2020**, *11*, 5959–5968.
- (43) Attwood, M. M.; Fabbro, D.; Sokolov, A. V.; Knapp, S.; Schiöth, H. B. Trends in kinase drug discovery: targets, indications and inhibitor design. *Nature Reviews Drug Discovery* **2021**, *20*, 839–861.
- (44) Frisch, M. J. et al. Gaussian~16 Revision C.01. 2016; Gaussian Inc. Wallingford CT.
- (45) Eberhardt, J.; Santos-Martins, D.; Tillack, A. F.; Forli, S. AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings. *Journal of Chemical Information and Modeling* **2021**, *61*, 3891–3898.
- (46) Ajileye, T.; Gainer, P.; Urbonas, M.; Pires, D. E. V. Automating reward function configuration for drug design. NeurIPS 2023 Workshop on New Frontiers of AI for Drug Discovery and Development. 2023.
- (47) Zhao, W. X. et al. A Survey of Large Language Models. *arXiv* **2023**,
- (48) Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; Polosukhin, I. Attention is All you Need. *Advances in Neural Information Processing Systems*. 2017.
- (49) Wang, C.; Wu, Q.; Weimer, M.; Zhu, E. FLAML: A Fast and Lightweight AutoML Library. *Proceedings of Machine Learning and Systems*. 2021; pp 434–447.
- (50) Bateman, A. et al. UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research* **2022**, *51*, D523–D531.
- (51) Davies, M.; Nowotka, M.; Papadatos, G.; Dedman, N.; Gaulton, A.; Atkinson, F.; Bellis, L.; Overington, J. P. ChEMBL web services: streamlining access to drug discovery data and utilities. *Nucleic Acids Research* **2015**, *43*, W612–W620.

- (52) McKinney, W. Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference. 2010.
- (53) mols2grid - Interactive molecule viewer for 2D structures. 2021; <https://github.com/cbouy/mols2grid>.

TOC Graphic

