

設定ファイルの共通行とコマンド履歴による生成 AI での 実習の質問メールの返信時間の短縮

有田 海斗¹ 平尾 真斗² 小山 智之² 串田 高幸¹

概要: 東京工科大学コンピュータサイエンス学部先進情報専攻では、先進情報プロジェクト実習 [IT・3] の実習が開かれている。実習を受ける学生は実習中に班のメンバと協力しながら実習課題に取り組み、実習課題を完了できなかった学生は翌週の実習までに宿題として実習課題を完了させる必要がある。学生は、実習の時間外で実習課題に分からないことがあった場合、実習で扱うサーバや実習課題の管理をしている TA に分からない内容について質問のメールを送信する。課題は、学生が質問メールを送信してから TA が返信を始めるまでに時間がかかるため、学生の学習が滞ることである。時間がかかる要因は、TA の他の授業の受講時間や研究室の作業時間により、すぐには返信できない時間が生じるからである。基礎実験として、実習で TA に送られたメールとその返信内容を集め、メールが返信されるまでにかかった時間を計測した。結果としては、学生が質問メールを送信してから返信されるまで 1 日以上時間がかかる場合は 21 件中 4 件の約 19%あり、2 日以上返信に時間がかかる場合が 21 件中 1 件の約 5%あるため、この学生の待ち時間を短縮する必要がある。提案手法は、質問メールの内容と実習を受けている全学生のコマンド履歴とファイルの内容を提案ソフトウェアに入力し、メールの返信を出力する。提案ソフトウェアは、ファイル特定フェーズ、初期プロンプト生成フェーズから生成 AI で返信内容と修正済みの質問者のファイルを作成し、出力結果評価フェーズで網羅率を使い評価を行う。出力結果評価フェーズで網羅率が 1.0 に満たない場合、改善プロンプト生成フェーズでプロンプトを再入力して再度評価を行い、網羅率が 1.0 になった時点で返信内容を出力する。評価方法は、2 つの指標を用いる。1 つ目は提案ソフトウェアの出力結果評価フェーズで網羅率が 1.0 になるまでに入力された改善プロンプトの回数であり、2 つ目は提案ソフトウェアが生成した返信内容と、TA が実際に作成した返信内容との意味的な類似性を Sentence BERT とコサイン類似度により数値化し評価を行う。

1. はじめに

背景

東京工科大学コンピュータサイエンス学部先進情報専攻では、先進情報プロジェクト実習 [IT・3] の実習が開かれている。この実習では、少人数の班による実習を行なうことによって、Site Reliability Engineering (以後 SRE とする) と関連する知識とスキルの習得を目的としている。SRE とは、大規模システムの信頼性と拡張性を確保することに重点を置いたソフトウェアエンジニアリングの分野である [1-3]。実習は毎週金曜日に 300 分間行われている。そして 7 人の Student Assistant (以後 SA とする) と、2 人の Teaching Assistant (以後 TA とする) が実習のサポートを行う。TA の 1 人はサーバの管理や実習課題の管理、

アナウンスやメール返信を行っている。

実習内では合計 30 台の Virtual Machine (以後 VM とする) があり、1 班から 10 班がそれぞれ 3 台ずつ VM を使っている。3 台の VM は全てに Ubuntu がインストールされており、Kubernetes のマスターノードの役割を持つ t サーバ、Kubernetes のワーカーノードの役割を持つ s サーバ、kubectrl コマンドで Kubernetes を制御する役割を持つ h サーバにそれぞれ分かれている。Kubernetes は、主要なオープンソースコンテナオーケストレーションプラットフォームである [4-7]。

学生は実習中に班のメンバと協力しながら実習課題に取り組む。学生は Docker の Image の作成や Kubernetes の Pod の作成を例とする SRE と関連したソフトウェアを使った実習課題が与えられる。Pod とは、Kubernetes における基本的なスケジューリングの最小単位であり、同時に実行される 1 つ以上のコンテナのグループを表す [8,9]。実習課題は班のメンバ全員が実習課題を完了し次第、次の

¹ 東京工科大学コンピュータサイエンス学部
〒 192-0982 東京都八王子市片倉町 1404-1

² 東京工科大学大学院バイオ・情報メディア研究科
〒 192-0982 東京都八王子市片倉町 1404-1

実習課題が公開されていき、実習課題を完了できなかった学生は翌週の実習までに宿題として実習課題を完了させる必要がある。宿題を完了した学生はメールで実習課題に取り組み、出力結果をスクリーンショットで撮影する。その後、スクリーンショットを Moodle のアップロード先に提出する。

学生は、実習の時間外で実習課題に分からないことがあった場合、実習で扱うサーバや実習課題の管理をしている TA に分からない内容について質問のメールを送信する。TA はメールの内容を確認し、学生が実習で使う Ubuntu に SSH して調査を行う。その後、調査結果を元に質問に返答する。

2025 年 6 月 9 日に 7 班の学生の 1 人が、実習課題 7-2 と 7-4 に取り組み、サーバに接続できないエラーが発生し、実習のメールサーバへ質問メールを送信した。実習課題 7-4 はランダムな変数を表示させる Web アプリケーションを K3s 上に作成する実習課題であり、質問者は実習課題 7-2 を参考に組み組んだ。実習課題 7-2 は Pod 名を表示する Web アプリケーションを K3s 上に作成する実習課題である。実際に送られたメールから、名前と学籍番号の個人情報を隠した質問された内容を内容 1 に、curl コマンドの出力結果を示した添付画像を文字に起こした内容を内容 2、Deployment の YAML 形式のファイルを示した添付画像を文字に起こした内容を内容 3 に示す。内容 1 では、課題 7-4 でエラーが出ており、課題を進めることができない。また、同じ内容の添付画像 2 を添付しており、添付する内容を間違えている。内容 2 では、curl コマンドで作成した Flask のコンテナを確認しようとしたが、Couldn't connect to server とエラーが出ており、sre007t の 31007 番ポートに接続できなかった。内容 3 では、質問者の Deployment の YAML 形式のファイルが添付されている。

また、質問者に対する TA の返信内容を内容 4 に示す。内容 4 では、TA は学生が書いた YAML 形式のファイルを確認し、YAML 形式のファイルと経験から設定されているポート番号が間違っていることに気がついた。その後、Port 番号を確認するように返信をした。

課題

課題は、学生が質問メールを送信してから TA が返信を始めるまでに時間がかかるため、学生の学習が滞ることである。学生が質問メールを送信してから TA が返信を行う過程について図 1 に示す。図 1 では、学生は実習用のメールサーバに質問メールを送信する。そして、TA が質問メールの確認を行い、返信内容作成を行う。その際、TA がメールを確認するまでにタイムラグが発生する。

タイムラグが発生する要因は、TA の他の授業の受講時間や研究室の作業時間により、すぐには返信できない時間が生じるからである。また、この実習では TA がメール返

内容 1: 名前と学籍番号の個人情報を隠した質問された内容

```
1 お忙しいところすみません。
2 (学籍番号)の(名前)と申します。この度は課題 7-4 についてご連絡させていただきました。
3 (添付画像 1)
4 質問というのが画像のようなエラーが出ることについてです。
5 いろいろと試行錯誤をしましたが、どうしてもうまくいきませんでした。なにが原因なのか教えて欲しいです。
6 (添付画像 2)
7 この画像はdeploymentの中身です。
8 講義資料の 7-2 を参考にしながら作成しました。
9 (添付画像 2)
10 続いてこちらがserviceの中身です。
11 こちらもさきほどと同様に課題 7-2 を参考にしながら作成しました。
12
13 かなり時間をかけて取り組みましたがどうしてもできません。
14 忙しいところ恐縮ですがよろしくお願いします。
```

内容 2: curl コマンドの出力結果を示した添付画像を文字に起こした内容

```
1 (学籍番号)sre007h:~/2025-06-06/simple-flask-read-write/k8s-file$
2 $ curl http://sre007t.local:31007/
3 curl: (7) Failed to connect to sre007t.local port 31007 after 107 ms: Couldn't connect to server
```

信をする時間が定められており、月曜日から金曜日の 9:00 から 17:00 までの時間帯のみ TA はメールの返信を行う。そのため、メール返信時間以外にメールが送信された場合、次の返信時間が来るまでメールの返信は行われない。

各章の概要

本稿は以下のように構成される。第 2 章では、本稿の関連研究について述べる。第 3 章では、本稿であげた課題を解決するための提案方式について述べる。第 4 章では、提案した手法の実装について述べる。第 5 章では、評価実験として実験内容と実験結果と分析について述べる。第 6 章では、提案手法についての議論を述べる。最後に、第 7 章にて結論を述べる。

2. 関連研究

構成ミスによるシステム障害を正常に動作している他のピアマシンと設定内容を比較することで、システム障害を発見、修正を行う研究がある [10]。この研究ではシステム障害を対処することを目的としている。本稿の背景であるプロジェクト実習 [IT・3] では、設定内容も実習を受ける

内容 3: Deployment の YAML 形式のファイルを示した添付画像を文字に起こした内容

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: (学籍番号)-flask-read-write-deployment
5  spec:
6    replicas: 3
7    selector:
8      matchLabels:
9        app: (学籍番号)-flask-read-write
10   template:
11     metadata:
12       labels:
13         app: (学籍番号)-flask-read-write
14     spec:
15       containers:
16         - name: (学籍番号)-flask-read-write
17           image: 192.168.50.244:5000/group07/(学籍番号)-flask-r
18           ports:
19             - containerPort: 32000
20       nodeSelector:
21         kubernetes.io/hostname: sre007t
```

内容 4: 質問者に対する TA の返信内容

```
1  TAの平尾です。
2  問題の件確認しました。コンテナ内で32000番でPortを
   公開して
3  いると思いますが、Kubernetes上で公開しているPort
   と一致
4  しているかどうかを確認してみてください。
```

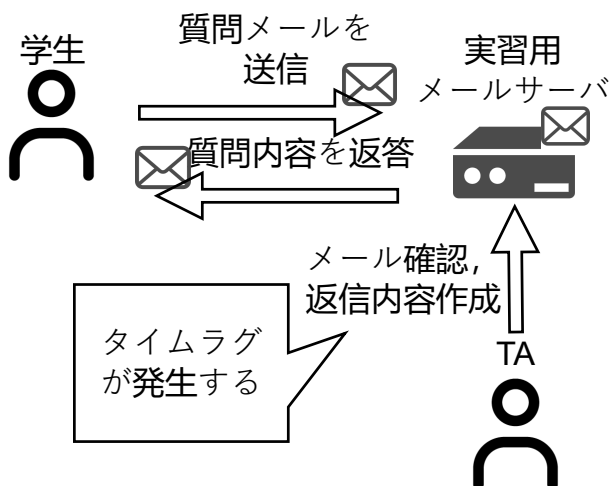


図 1: 学生が質問メールを送信してから TA が返信を行う過程

学生間でそれぞれ少しずつ異なっており、正常に動作しているサーバが一意に定まっていないため、同じ提案手法を使用しシステム障害の特定をするために必要な情報が不足

している。

ソフトウェアの障害発生箇所の特定に使われているスペクトラムベースの位置特定手法を複雑なネットワーク障害の障害発生箇所の発見にも応用した研究がある [11]. この研究ではテストケースを元にネットワークの設定を 1 行ずつ確認し、その行に suspiciousness score をつけることで原因特定を行っている. 本稿の背景であるプロジェクト実習 [IT・3] では、テストケースが存在せず、正解となる動作がインプットされていないため、同じ提案手法を使用しソフトウェア障害の特定をするために必要な情報が不足している.

大規模言語モデルを使い、Kubernetes の YAML 形式のファイルのミス構成の検出、位置特定、推論および修復を行う研究がある [12]. この研究では Kubernetes の YAML 形式のファイルのみを対象としているが、本稿の背景であるプロジェクト実習 [IT・3] では、YAML 形式のファイルのミス構成の検出は行うが、YAML 形式のファイルの推論および修復を行うだけでは学生の学習のためにならないため、学生の調査ステップに合わせて質問メールに返信をする.

3. 提案

本稿では、実習を受ける学生の質問メールを受け取り、メールの返信を出力する提案手法について示す. 実習課題に取り組んでいるときにエラーが出た場合、`kubectl get` リソース名でステータスの確認や `kubectl logs` Pod 名でログの確認を行い、エラーの原因を調査してそれに応じた解決策を実行する. そのため、学生が課題を解決するための調査ステップに合わせて質問への返信を行う手法を提案する.

提案方式

提案手法は、質問メールの内容と実習を受けている全学生のコマンド履歴とファイルの内容を提案ソフトウェアに入力し、メールの返信を出力する. 提案ソフトウェアは、ファイル特定フェーズ、初期プロンプト生成フェーズ、出力結果評価フェーズの 3 つのフェーズと、出力結果評価フェーズの評価が振るわない場合に改善プロンプト生成フェーズの 1 つのフェーズを経て、メールの返信内容を作成する. 提案手法の全体の構成図を図 2 に示す. ファイル特定フェーズでは、質問者のメールの内容から質問者が何を対象に質問を行っているのかファイルを特定し、(1) のファイルパスとファイル内容を初期プロンプト生成フェーズに出力する. 初期プロンプト生成フェーズでは、ファイルパスとファイル内容と全学生のコマンド履歴から質問者と質問者以外の学生の調査ステップの一致率を比較し、一致率に応じて (2) で生成 AI に初期プロンプトとして出力する. その後、(3) で生成 AI から生成 AI の出力を出力結

果評価フェーズに出力する。出力結果評価フェーズでは、全学生のファイルと生成 AI の出力である修正済みの質問者のファイルを使い、(4) で評価が基準に満たないなら改善プロンプト生成フェーズに修正済みの質問者のファイルや評価データを出力する。改善プロンプト生成フェーズでは、(5) で改善プロンプトを生成 AI に送り、(6) で改善プロンプトを送った生成 AI の再出力を受け取る。その後、生成 AI に再出力された結果を (7) で出力結果評価フェーズに戻り再評価する。最後に出力結果評価フェーズで (8) の評価基準を満たしていた場合、(9) でメールを送信する。

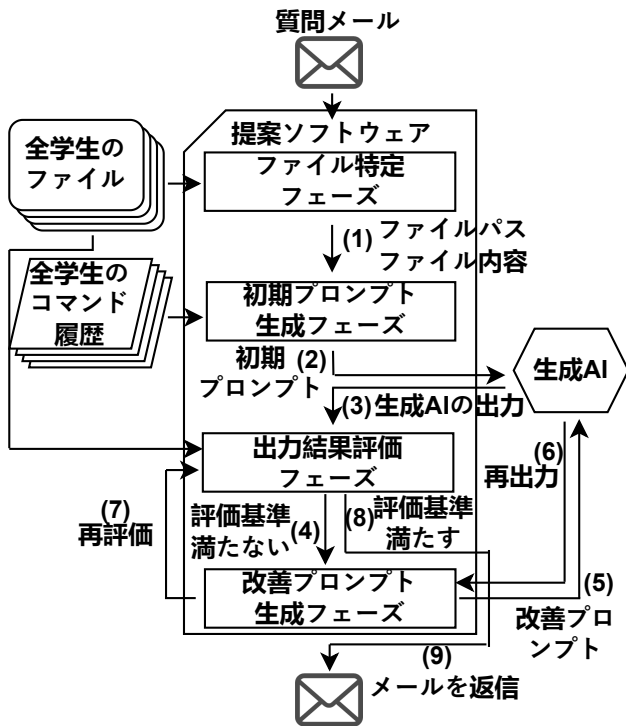


図 2: 提案方式の全体の構成図

ファイル特定フェーズ

ファイル特定フェーズでは、質問メールの文面から対象となるファイルを特定する。本フェーズでは、.yaml や .py を例とする文中に含まれる拡張子を持つ単語を抽出し、それがどの実習課題に関連しているかを明らかにする。例えば、メール内のファイル名表記 .yaml や第 n-n 回の課題番号に基づいて、実習課題の PDF 資料に記載されたファイルと照合を行う。実習課題の PDF 資料との照合は、あらかじめ提案ソフトウェアに実習課題の PDF 資料を保存しておき、拡張子を持つ単語を PDF 資料から検索させる。これにより、質問者がどのファイルを対象にしているかを明示的に特定し、以降のフェーズで用いる対象ファイルを決める。内容 1 では、メール内容に記載された課題 7-2 と 7-4 に使うファイルを探す。講義資料の PDF から拡張子で検索を行うと、flask-deployment-mod.yaml と

flask-service-mod.yaml, read-write.py が見つかる。

初期プロンプト生成フェーズ

初期プロンプト生成フェーズでは、質問メールの内容とファイル特定フェーズで特定したファイルとコマンド履歴を使い、質問者の調査ステップに応じた生成 AI の初期プロンプトを生成する。初期プロンプト生成フェーズにおける生成 AI への入力と出力を表した図を図 3 に示す。図 3 では、実習に参加している全学生のコマンド履歴と質問メール、質問者のファイルを入力する。その後、全学生のコマンド履歴から質問者のコマンド履歴と調査用のコマンド履歴に分ける。質問者のコマンド履歴と調査用のコマンド履歴から一致率を出し、一致率が 50%未満のとき、50%以上 80%未満のとき、80%以上のとき、それぞれでプロンプトテンプレートからプロンプトを選択する。基本プロンプトと質問メール、質問者のファイルを初期プロンプトとして生成 AI に入力して、生成 AI の出力を出力結果評価フェーズに送る。

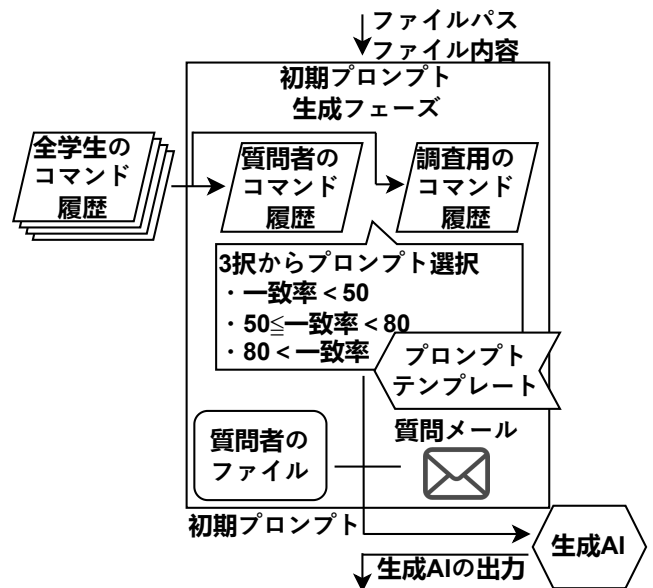


図 3: 初期プロンプト生成フェーズにおける生成 AI への入力と出力を表した図

全学生のコマンド履歴から質問者のコマンド履歴と調査用のコマンド履歴に分ける方法を示す。はじめに、`kubectl apply -f ファイル名のコマンド`を実行してから、再び `kubectl apply -f ファイル名`を実行するまでのコマンドを収集する。収集する `kubectl apply -f ファイル名`のコマンドから、過半数以上が実行するコマンドを調査用のコマンドとして収集する。ただし、Pod 名の違いや学籍番号の違いを考慮し、75%以上一致している場合は同一コマンドとして収集する。

内容 1 の質問メールにおける一致率の出し方を、他

の 3 人の学生のメールから作成する例を示す。内容 1 における学生の入力したコマンド例 1 を内容 5, 学生の入力したコマンド例 2 を内容 6, 学生の入力したコマンド例 3 を内容 7 に示す。これらのコマンド履歴は, flask-deployment-mod.yaml を apply してから再び apply するまでのコマンド履歴である。内容 5 は, kubectl get pods と kubectl get deployment でステータスを確認し, vi flask-service-mod.yaml で次の YAML 形式のファイルを作成し, apply している。内容 6 では, 複数回 kubectl get pods でステータスを確認をし, kubectl describe pods (学籍番号)-flask-deployment-mod-6fd968fd9d-5lsp4 で詳細情報を確認している。最後に vi flask-deployment-mod.yaml で apply 前と同じファイルを編集し, 再び apply を行っている。内容 7 は, kubectl get pods と複数回の kubectl get deployment でステータスを確認し, vi flask-deployment-mod.yaml でファイルを確認している。もう一度 k get pods でステータスを確認した後, kubectl logs (学籍番号)-flask-deployment-ffc68c457-5pgf8 でログの確認, kubectl describe pods (学籍番号)-flask-deployment-ffc68c457-5pgf8 で詳細情報を確認している。最後に vi flask-service-mod.yaml と vi flask-deployment-mod.yaml でファイルの編集を行い, 再び apply を行っている。

内容 5: 学生の入力したコマンド例 1

```
1 kubectl apply -f flask-deployment-mod.yaml
2 kubectl get pods
3 kubectl get deployments
4 vi flask-service-mod.yaml
5 kubectl apply -f flask-service-mod.yaml
```

内容 6: 学生の入力したコマンド例 2

```
1 kubectl apply -f flask-deployment-mod.yaml
2 kubectl get pods
3 kubectl get pods
4 kubectl get pods
5 kubectl get pods
6 kubectl get pods
7 kubectl describe pod (学籍番号)-flask-deployment-mod-6fd968fd9d-5lsp4
8 vi flask-deployment-mod.yaml
9 kubectl apply -f flask-deployment-mod.yaml
```

これらのコマンド履歴から調査用のコマンドを求めるため, 一致率を使う。一致率を式 (1) に示す。式 (1) では, m を一致率, s を学生数, t を学生の総数とする。学生数と

内容 7: 学生の入力したコマンド例 3

```
1 kubectl apply -f flask-deployment-mod.yaml
2 kubectl get deployments
3 kubectl get deployments
4 kubectl get deployments
5 kubectl get pods
6 kubectl get services
7 vi flask-deployment-mod.yaml
8 k get pods
9 kubectl logs (学籍番号)-flask-deployment-ffc68c457-5pgf8
10 kubectl describe pod (学籍番号)-flask-deployment-ffc68c457-5pgf8
11 vi flask-deployment-mod.yaml
12 vi flask-service-mod.yaml
13 kubectl apply -f flask-deployment-mod.yaml
```

は, 対象のコマンドを実行した学生の人数である, 学生の総数とは, 提案ソフトウェアがコマンド履歴を参照した学生の人数である。

$$m = \frac{s}{t} \quad (1)$$

内容 5, 内容 6, 内容 7 におけるそれぞれのコマンドの学生数と一致率を表 1 に示す。コマンドの列では, それぞれの内容の学生の全コマンドが入っている。内容 6 の kubectl describe pods (学籍番号)-flask-deployment-mod-6fd968fd9d-5lsp4 と内容 7 の kubectl describe pods (学籍番号)-flask-deployment-ffc68c457-5pgf8 は, コマンドの単語が 75%以上一致しているため, 同じコマンドとみなしてカウントしている。学生数の列では, それぞれの内容でコマンドを使っている学生の人数をカウントしている。例えば, kubectl get deployment は, 内容 5 と内容 7 で使われているため, 2 となる。一致率の列では, 式 (1) で求める, 学生の総数 t は 3 とする。

表 1: 内容 5, 内容 6, 内容 7 におけるそれぞれのコマンドの学生数と一致率

コマンド	学生数	一致率
kubectl get pods	3	3/3=1.0
kubectl get deployments	2	2/3=0.7
vi flask-service-mod.yaml	2	2/3=0.7
kubectl describe pod (学籍番号)-flask-deployment-mod-6fd968fd9d-5lsp4	2	2/3=0.7
kubectl describe pod (学籍番号)-flask-deployment-ffc68c457-5pgf8	2	2/3=0.7
vi flask-deployment-mod.yaml	2	2/3=0.7
kubectl get services	1	1/3=0.3
kubectl logs (学籍番号)-flask-deployment-ffc68c457-5pgf8	1	1/3=0.3

次に、式 (1) で求めた一致率を使い、プロンプトを選択する。一致率が 50%未満の場合、質問者に対し不足している調査用のコマンドを提示し、まずその出力を確認するよう促す。一致率が 50%以上 80%未満の場合、ログの断片やエラー箇所を例とする特に注目すべき情報を指摘するようなプロンプトを生成 AI に与える。一致率が 80%以上であれば、調査ステップを十分に踏んでいると見なし、直接的にエラー原因や修正箇所を提示するようなプロンプトを生成 AI に与える。具体的には、一致率が 50%未満の場合、「以下は先進情報プロジェクト実習 [IT・3] に来た質問メールです。調査方法を指摘してください。返信内容だけを出力してください。以下は他の学生が試している調査方法です。」の後ろに質問者と質問者以外で同一ではなかったコマンドを加えて、基本プロンプトとする。一致率が 50%以上 80%未満の場合、「以下は先進情報プロジェクト実習 [IT・3] に来た質問メールです。特に確認すべき項目を指摘するように返信内容を指摘してください。返信内容だけを出力してください。」を基本プロンプトとする。一致率が 80%以上であれば、「以下は先進情報プロジェクト実習 [IT・3] に来た質問メールです。直接的に答えを解説するように指摘してください。返信内容だけを出力してください。」を基本プロンプトとする。最後に、基本プロンプトに加えて、「以下はメール内容です。」の後ろに質問者のメール内容を加え、「以下は質問者のファイルです。」の後ろに質問者のファイルを加えることで、初期プロンプトとする。

内容 1 では、調査用のコマンドは `kubectl get deployments`, `kubectl get pods`, `vi flask-service-mod.yaml`, `kubectl describe pod` Pod 名の 4 つであり、質問者は調査用のコマンドのうち `kubectl get pods` と `kubectl get deployment` の 2 つが実行されていた。そのため、一致率は 2/4 で計算を行い、50%となり、ログの断片やエラー箇所を例とする特に注目すべき情報を指摘するようなプロンプトを作成する。

内容 1 で生成 AI へ入力するプロンプトを内容 8 に示す。内容 8 では、一致率が 50%以上 80%未満の際のプロンプトを基本プロンプトに入力する。基本プロンプトは、「以下は先進情報プロジェクト実習 [IT・3] に来た質問メールです。特に確認すべき項目を指摘するように返信内容を指摘してください。返信内容だけを出力してください。」と入力する。(質問者のメールの内容)に内容 1 と内容 2 と内容 3、(質問者のファイル)で `flask-deployment-mod.yaml` と `flask-service-mod.yaml`, `read-write.py` を初期プロンプトとして入力する。

内容 8: 生成 AI へ入力するプロンプト

- 1 (基本プロンプト)
- 2
- 3 (質問者のメールの内容)

- 4
- 5 (質問者のファイル)

そして、内容 8 を初期プロンプトとして入力した生成 AI の出力を内容 9 に示す。内容 9 では、レプリカ数、Pod の状態、Service の状態、ファイアウォールやセキュリティグループ、ノードの状態を確認し、再度状況を確認するように促す生成 AI の出力がでた。このプロンプトを評価し、評価基準を満たした場合、内容 9 が返信内容となる。

内容 9: 生成 AI の出力結果

- 1 お疲れ様です。課題 7-4 についてのご質問、ありがとうございます。
- 2 お送りいただいた情報から、特に以下の点について確認をお願いいたします。
- 3 Deploymentのreplicas数とServiceのPodセクタの整合性: `flask-read-write-deploy.yaml`ではreplicas: 1と設定されていますが、画像(`image_d58621.png` と `image_d58620.png`)ではreplicas: 3と表示されています。どちらが現在の正しい設定でしょうか？
- 4 また、Serviceのselector: app: (学籍番号)-flask-read-writeとDeploymentのtemplate.metadata.labels: app: (学籍番号)-flask-read-writeは一致しており問題ありません。
- 5 Serviceのポート設定とアクセスURL:
- 6 Service (`flask-read-write-svc.yaml`) ではnodePort: 31007, port: 31007, targetPort: 32000と設定されています。
- 7 `read-write.py`ではFlaskアプリケーションがport=32000で待ち受けています。
- 8 `curl`コマンドでは`http://sre007t.local:31007/`にアクセスされており、これはServiceのNodePortに合致しています。
- 9 `curl`のエラーは「`Couldn't connect to server`」となっており、
- 10 これはネットワーク的な接続が確立できていないことを示唆しています。
- 11 Podの状態確認:
- 12 Deploymentが正常にデプロイされ、Podが起動しているか確認してください。以下のコマンドでPodの状態を確認できます。
- 13 Bash
- 14 `kubectl get pods -l app=c0a23073f3-flask-read-write`
- 15 もしPodがRunning状態でない場合、`kubectl describe pod <Pod名>`や`kubectl logs <Pod名>`で詳細なエラーを確認してください。
- 16 Serviceの状態確認:
- 17 Serviceが正しく作成され、エンドポイントが割り当てられているか確認してください。
- 18 Bash
- 19 `kubectl get svc flask-read-write-svc`
- 20 `kubectl describe svc flask-read-write-svc`
- 21 `kubectl describe svc`の出力で、Endpointsの項目にPodのIPアドレスとtargetPortが正しく表示されているか確認してください。
- 22 `firewall / セキュリティグループの確認:`

```

23 sre007t.localのファイアウォールやセキュリティグル
    ープの設定で、ポート 31007へのインバウンド接続が許
    可されているか確認してください。Kubernetesクラスタ
    内外からのアクセスを妨げている可能性があります。
24 Nodeの状態確認:
25 対象のノード (sre007t)が正常に稼働しているか確認
    してください。
26 Bash
27 kubectl get nodes
28 上記項目をご確認いただき、再度状況をお知らせいた
    だけますでしょうか。
29 よろしくお願いたします。
    
```

出力結果評価フェーズ

出力結果評価フェーズでは、生成 AI が出力した内容が適切かどうかを定量的に評価する。出力結果評価フェーズにおける入力と出力を表した図を図 4 に示す。

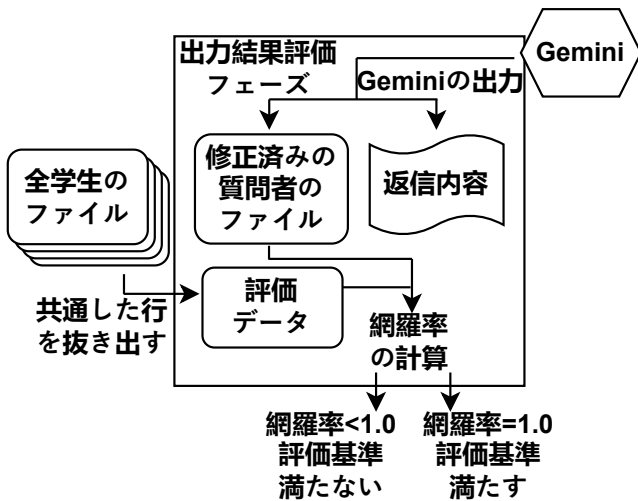


図 4: 出力結果評価フェーズにおける入力と出漁を表した図

図 4 では、Gemini は提案手法で作成した返信内容に加えて、修正済みの質問者のファイルも出力する。内容 1 で生成 AI が作成した修正済みの質問者のファイルを内容 10 に示す。内容 10 では、生成 AI が名前やレプリカ数を修正した後の Deployment の YAML 形式のファイルが出力できる。

内容 10: 生成 AI が作成した修正済みの質問者のファイル

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: (学籍番号)-flask-read-write-deploy # 現
    在は末尾に'-deploy'がついている
5  spec:
6    replicas: 1 # 質問元の画像では 3でしたが、提供さ
    れたテキストでは 1
7    selector:
8      matchLabels:
    
```

```

9    app: (学籍番号)-flask-read-write
10   template:
11     metadata:
12       labels:
13         app: (学籍番号)-flask-read-write
14     spec:
15       containers:
16         - name: flask-read-write-container # 現
    在は'flask-read-write-container'
17         image: 192.168.50.244:5000/group07/(学
    籍番号)-flask-read-write
18         ports:
19         - containerPort: 32000
20     nodeSelector:
21       kubernetes.io/hostname: sre007t
    
```

次に全学生の同名ファイルを集集し、各ファイルの中に共通して含まれている行を抜き出し、評価データとして抽出する。内容 1 の flask-deployment-mod.yaml における評価データを、内容 11 に示す。内容 11 では、完全に一致している行のみを抽出するため、ポート番号や名前 image 名、ホスト名、レプリカ数は削除されている。この抜き出された行が評価データとして使用される。

内容 11: 内容 1 の flask-deployment-mod.yaml における評価データ

```

1  apiVersion:apps/v1
2  containers:
3  kind:Deployment
4  labels:
5  matchLabels:
6  metadata:
7  nodeSelector:
8  ports:
9  selector:
10 spec:
11 template:
    
```

その後、修正済みの質問者のファイルがファイルを作成する上で評価データをどのくらい含んでいるのかを測定するために網羅率を出す。c を網羅率、n を評価データ、r を修正済みの質問者のファイルの行として、網羅率を式 (2) に示す。網羅率は、評価データと修正済みの質問者のファイルの行の共通部分を評価データで割ったものである。網羅率が 1.0 未満なら評価基準を満たさなかったとして、網羅率が 1.0 であるなら評価基準を満たしているとする。網羅率により Gemini の出した答えの正確さを確認できる。

$$c = \frac{|n \cap r|}{|n|} \quad (2)$$

例えば、内容 10 と内容 11 の網羅率は、評価データ n の 11 行を修正済みの質問者のファイル r が 11 行全てを含んでいるため、11/11 で 1.0 である。網羅率が 1.0 であるの

で、生成 AI が出力した内容が適切だと評価をし、返信メールで返答を行う。

改善プロンプト生成フェーズ

改善プロンプト生成フェーズでは、出力結果評価フェーズにおいて基準を満たさなかった生成 AI の出力に対して、その精度を向上させるための追加プロンプトを生成する。出力結果評価フェーズにて、評価データを明らかにした。これらの評価データのうち、生成 AI が出力した修正済みの質問者のファイルの行に含まれていなかった行を抽出し、それらを明示的に含むよう促す改善用プロンプトを作成する。この改善用プロンプトを用いて再度生成 AI に出力を行わせ、もう一度評価を行う。網羅率が 1.0 になった時点で、その出力結果を質問者への回答とする。プロンプトの出力結果からプロンプトの改善の反復を行い、LLM の出力を改善する研究が行われている [13]。その研究では、自動的な反復回数が 2 回から 3 回で改善のピークとなっている。そのため、3 回以上繰り返された場合、これ以上の改善は諦め、調査用のコマンドを他の学生が入力していたコマンドとして質問メールの返答に付ける。この改善を行い、質問者への回答が間違った修正を行う指摘をしないようにする。

内容 8 では、生成 AI が出力した内容が適切だと評価されたため、改善プロンプト生成フェーズは実行しない。もし不適切だと評価された場合は、内容 11 を追加プロンプトとして入力する。

ユースケース・シナリオ

本稿では、東京工科大学の先進情報プロジェクト実習 [IT・3] をユースケースとしている。この実習では、1 班から 10 班がそれぞれ VM を 3 台ずつ使用しており、Kubernetes のマスターノードの役割を持つ t サーバ、Kubernetes のワーカーノードの役割を持つ s サーバ、kubectl コマンドで Kubernetes を制御する役割を持つ h サーバにそれぞれ分かれている。そして、1 から 10 の t サーバ、s サーバ、h サーバはポート番号やホスト名を除いて同じ設定が行われている。学生はそのサーバを使い、学生同士で協力しながら同じ実習課題に取り組むため、実習課題に取り組んだ後もサーバ内の設定はホスト名やポート番号を除いて同じ設定が行われる。

実習の時間中に実習課題を完了できなかった学生は、翌週の実習までに宿題として実習課題を完了させる。学生は、実習の時間外に実習課題に取り組む過程で分からないことがあった場合、TA に分からない内容について質問メールを送信する。学生は実習課題に取り組む中で分からないことを質問するために、メールを実習用のメールサーバに送信する。提案ソフトウェアは、メールを受け取った実習用のメールサーバからメールを受信する。返信内容を作成し

た提案ソフトウェアは、実習用メールサーバにメールを返信する。作成した学生の調査ステップに合わせたメールを返信する。

4. 実装

実装では提案手法をもとにソフトウェアを作成する。ソフトウェアの作成には、開発言語である Python3 を用いる。開発するソフトウェア内のプログラムがどのような機能を持っているのか説明する。ソフトウェアの全体像を図 5 に示す。図 5 では、はじめに学生が質問メールを送信する。そのメールを提案ソフトウェアが実習用メールサーバに取得する。次に 1 班サーバから 10 班サーバまでのコマンド履歴とファイルの内容を提案ソフトウェアに入力する。提案ソフトウェアは生成 AI である Gemini にプロンプトを入力し、その出力を提案ソフトウェアに返す。最後に提案ソフトウェアが実習用メールサーバにメールを送信し、実習用メールサーバから学生へ調査ステップに合わせた返信が返ってくる。

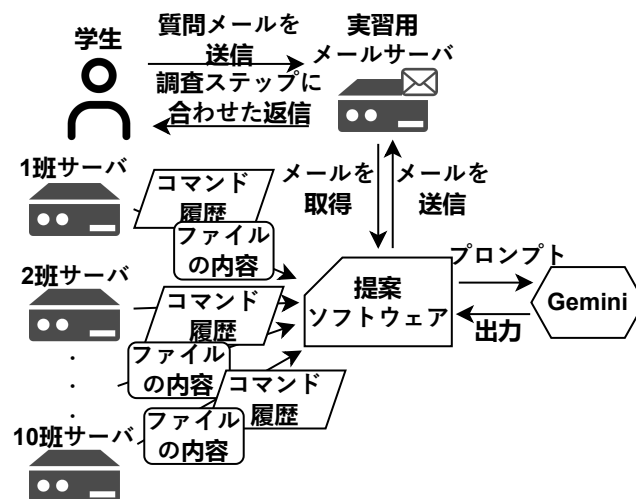


図 5: ソフトウェアの全体像

提案ソフトウェアは 5 つの機能を持っている。1 つ目は、質問メールの未読を確認しファイルパスを特定する機能である。2 つ目は、全ての班の全学生のコマンド履歴とファイルの内容を収集する機能である。3 つ目は、プロンプトを作成し Gemini に入力する機能である。4 つ目は、Gemini の出力結果を評価する機能である。5 つ目は、質問者にメールで返信する機能である。

質問メールの未読を確認しファイルパスを特定する機能

この機能では、はじめに実習用メールサーバに届いた質問メールの中から、まだ読まれていないメールを取得する。その後、メール内容から、.yaml や .py を例とする拡張子を含んでいるファイル名を探し出す。さらに、メール内容から数字-数字や数字-数字の文字列を探す。この文字列

を実習の課題番号として、あらかじめ提案ソフトウェアに保存した実習課題の PDF 資料から .yaml や .py を例とする拡張子を含んでいるファイル名を探し出す。その後、find コマンドをホームディレクトリから実行することで、その質問がどのファイルに関する内容なのかを特定し、そのファイルパスを出力する。

全ての班の全学生のコマンド履歴とファイルの内容を収集する機能

この機能では、1 班から 10 班のサーバの Ubuntu に SSH を行い、コマンド履歴とファイルの内容を取得する。ファイルの内容の取得には、前述の機能で特定したファイルパスのユーザー名を変更したパスを使用する。また、提案ソフトウェアは全学生の .bash_history ファイルを取得し、コマンド履歴を集める。

プロンプトを作成し Gemini に入力する機能

この機能では、集めた全学生のコマンド履歴とファイルの内容とメール内容から初期プロンプトを作成し、Gemini に入力する。Gemini とは、Google が開発した大規模言語モデルと自然言語処理を備えたマルチモーダル AI ツールである [14]。Gemini の API は無料枠が設けられており、小規模な生成 AI の利用に向いている。はじめに、コマンド履歴を質問者と質問者以外の学生のコマンド履歴に分ける。次に質問者以外の学生のコマンド履歴から、質問対象のファイルを kubectl apply してから別のファイルを apply するまでのファイルを一時ファイルとして保存する。その一時ファイルから過半数よりも多く同じ単語が続いているコマンドを同じコマンドとみなし、異なる単語同士からランダムに 1 単語を決め、同じ単語に置き換え一般化する。単語は、コマンドをスペースで区切ったものを単語とする。その後、一般化した質問者と質問者以外の学生のコマンドから、同一のコマンドを抜き出し、一致率を求める式 (1) を使い一致率を求める。一致率が 50% 未満の場合、「以下は先進情報プロジェクト実習 [IT・3] に来た質問メールです。調査方法を指摘してください。返信内容だけを出力してください。以下は他の学生が試している調査方法です。」の後ろに質問者と質問者以外で同一ではなかったコマンドを加えて、基本プロンプトとする。一致率が 50% 以上 80% 未満の場合、「以下は先進情報プロジェクト実習 [IT・3] に来た質問メールです。特に確認すべき項目を指摘するように返信内容を指摘してください。返信内容だけを出力してください。」を基本プロンプトとする。一致率が 80% 以上であれば、「以下は先進情報プロジェクト実習 [IT・3] に来た質問メールです。直接的に答えを解説するように指摘してください。返信内容だけを出力してください。」を基本プロンプトとする。最後に、基本プロンプトに加えて、「以下はメール内容です。」の後ろにメール内容

を加え、「以下は質問者のファイルです。」の後ろに質問者のファイルを加えることで、初期プロンプトとする。その初期プロンプトを Gemini API を使い入力する。

Gemini の出力結果を評価する機能

この機能では、Gemini API の出した出力結果を評価し、評価結果によって返信をするか決める機能である。生成 AI に追加で、「修正済みの質問者のファイルだけを出力してください。」とプロンプトに入力し、その出力結果を修正済みの質問者のファイルとして保存する。次に、全学生のファイルを確認し、同じ行が書かれている行を全ての学生の人数だけループを回して見つけ、評価データとしてテキストファイルに保存する。その後、修正済みの質問者のファイルから、評価データを抜き出し、網羅率を求める式 (2) を使い網羅率を求める。網羅率が 1.0 であれば質問者にメールを送信できるとし、次の質問者にメールで送信する機能に進む。1.0 に満たない場合は、追加のプロンプトとして、評価データを追加のプロンプトとして入力する。その上で同じように網羅率を求め評価を行い、網羅率が 1.0 であれば質問者にメールを送信できるとし、次の質問者にメールで送信する機能に進む。それを最大 3 回まで繰り返した場合、「以下が他の学生が問題を解決するために入力していたコマンドです。」の後ろにプロンプトを作成し Gemini に入力する機能で求めた全学生で同一のコマンドを加え、返信内容とする。

質問者にメールで返信する機能

この機能は、Gemini の出力結果を評価する機能で評価を満たした修正済みの質問者のファイルに紐づいた返信内容を質問者に返信する機能である。返信内容を決定したら、Gmail API を使い、質問者のメールアドレス宛に返信を行う。タイトルは Re: 質問者のメールのタイトルとし、本文は Gemini の出力そのままを本文とする。その後、未読メールに既読をつけ、プログラムを終了させる。

5. 評価方法

本稿では、提案ソフトウェアの出力結果の有効性を評価するために、以下の 2 つの指標を用いる。1 つ目は、提案ソフトウェアの出力結果評価フェーズで網羅率が 1.0 になるまでに入力された改善プロンプトの回数である。改善プロンプトの入力回数は、提案ソフトウェアが生成 AI を呼び出している回数である。生成 AI を呼び出す回数が増えると、メールの返信に時間がかかり、トークン消費量が増えるため、性能の低さを評価する指標になる。

2 つ目は、類似度スコアによる自動評価である。この指標では、提案ソフトウェアが生成した返信内容と、TA が実際に作成した返信内容との意味的な類似性を自然言語処理により数値化し、自動で評価を行う。評価には、それぞれの

返信内容を文書ベクトルとして表現し、コサイン類似度によりスコアを算出する。ベクトル化には、Sentence BERT を使いベクトル化する。Sentence BERT とは、BERT を改良したベクトル化の手法であり、シャムネットワーク構造とトリプレットネットワーク構造を使用してより高速な文章同士の比較をすることができる [15]。コサイン類似度を式 (3) に示す。 x は TA の返信内容、 y は提案ソフトウェアの返信内容をそれぞれベクトル化したものとする。

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|} \quad (3)$$

これら 2 つの指標により、提案ソフトウェアが実習支援において有効に機能しているかを定量的に評価する。

実験環境

プロジェクト実習 [IT・3] で使用している環境である 1 班から 10 班の学生がファイルを作成したりコマンドを主に実行している VM を使って実験を行う。それぞれの VM の構成は、vCPU が 2Core、メモリが 8GB、ディスクが 50GB、OS は Ubuntu 24.04.2 を使用している。TA・SA の実験用の VM に Python3 の仮想環境を作成し、提案ソフトウェアを実行する。

基礎実験

基礎実験として、先進情報プロジェクト実習 [IT・3] において TA に送られたメールとその返信内容を集め、メールが返信されるまでにかかった時間を計測し、その結果を示す。第 1 回の実習である 2025 年 4 月 18 日から第 12 回の実習である 2025 年 7 月 11 日の期間で送られた、21 件のメールを対象とする。TA がメール返信をするまでにかかった時間を表すグラフを図 6 に示す。横軸は TA がメールの回答にかかった時間を分単位で表しており、横軸は質問メールの回数を表す。結果は、0 分以上 500 分未満の時間に返信できた件数が 10 件、500 分以上 1000 分未満に返信できた件数が 4 件、1000 分以上 1500 分未満に返信できた件数が 3 件、1500 分以上 2000 分未満に返信できた件数が 1 件、2000 分以上 2500 分未満に返信できた件数が 2 件、3500 分以上 4000 分未満に返信できた件数が 1 件の結果となった。そのため、1 日以上返信に時間がかかる場合が 21 件中 4 件の約 19%あり、2 日以上返信に時間がかかる場合が 21 件中 1 件の約 5%あった。この結果から、先進情報プロジェクト実習 [IT・3] では、学生が質問メールを送信してから返信されるまで時間がかかる。

6. 議論

本稿の提案手法では、コマンド履歴の一致率が 50%未満、50%以上 80%未満、80%以上でそれぞれ生成 AI に入力するプロンプトが変更され、履歴との一致度が高いほど、より具体的な指示を含んだプロンプトが使用されるように

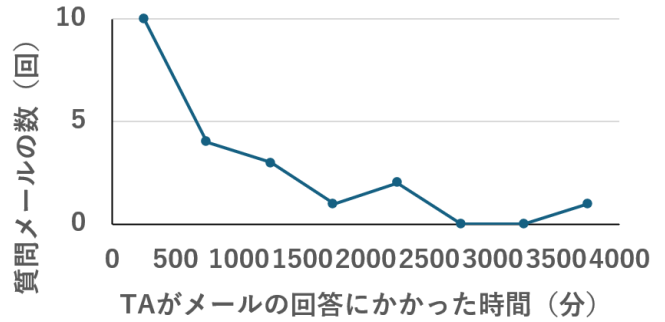


図 6: TA がメール返信をするまでにかかった時間を表すグラフ

意図して設計した。これら 3 つの条件は、実習の SA をした経験則から理解度を 3 段階にわけて設定している。そのため、これらの閾値が実際の学習支援において最適であるかどうかについては基礎実験をする必要がある、具体的には、プログラム適用後の学生からの再質問回数とコマンドの種類を測定し、最も効果が高い閾値に合わせていく必要がある。

本稿の提案手法では、コマンド履歴の一致率に応じたプロンプトの選択と生成 AI の出力の評価により、質問者のエラーを調査した量に応じた返答を返すことに主眼をおいている。一方で、エラーを解決する力は生成 AI に依存しており、プロンプトの設計も最適化されていない。これを解決する方法として、生成 AI の参照可能な知識源を拡張し、より文脈に即した情報を提供する仕組みの導入が有効である。講義資料をベクトル化し、質問に含まれる言葉やファイルと意味的に近い情報を提示する仕組みを統合することで、エラーに関連する文脈と類似度の高い情報を活用できる仕組みを導入する。

本稿の提案手法では、生成 AI の出力を評価する方法として、出力結果評価フェーズで返信内容そのものではなく、修正済みの質問者のファイルの評価している。しかし、実際に質問者に出力するのは修正済みの質問者のファイルではなく、返信内容を出力する。そのため、返信内容自体を提案ソフトウェアが評価できる仕組みが必要である。方法として、返信内容が最低限含むべきキーワードを定義し、それを満たしているかを自動で判定する方法がある。Kubernetes のエラーの原因を特定する研究が行われており、エラー原因を特定する手法を提案ソフトウェアに組み込む [16,17]。それによりエラー原因箇所を特定し、エラー箇所のリソース名やラベルをキーワードとして設定する。このキーワードを返信内容が含んでいるのかで評価を行う。

本稿の提案手法では、網羅率が 1.0 になるまで改善プロンプト生成フェーズが最大 3 回呼び出される。3 回繰り返す根拠は関連研究における LLM の改善回数であり、本稿の提案手法のプロンプトにも有効か実験が必要である。実験では、改善プロンプト生成フェーズにてプロンプトを繰り返

返す回数を初期プロンプト生成の1回と合わせて、Gemini 2.5 Pro の API の無料枠で1日に利用できる100回とする。100回のうち、4回連続で出力結果評価フェーズの網羅率が変化しなくなるまで実験をおこなう。4回連続で網羅率が一致する確率は1/16で約6%である。その結果を改善プロンプト生成フェーズを繰り返す回数とする。

7. おわりに

先進情報プロジェクト実習 [IT・3] の実習にきた質問のメールに TA が返信を始めるまでに時間がかかるため、自動返信ソフトウェアで時間を短縮することを目的としている。基礎実験として、実習で TA に送られたメールとその返信内容を集め、メールが返信されるまでにかかった時間を計測した。結果としては、学生が質問メールを送信してから返信されるまで1日以上時間がかかる場合は21件中4件の約19%あり、2日以上返信に時間がかかる場合は21件中1件の約5%あるため、この学生の待ち時間を短縮する必要がある。提案手法は、質問メールの内容と実習を受けている全学生のコマンド履歴とファイルの内容を提案ソフトウェアに入力し、メールの返信を出力する。提案ソフトウェアは、ファイル特定フェーズ、初期プロンプト生成フェーズから生成 AI で返信内容と修正済みの質問者のファイルを作成し、出力結果評価フェーズで網羅率を使い評価を行う。出力結果評価フェーズで網羅率が1.0に満たない場合、改善プロンプト生成フェーズでプロンプトを再入力して再度評価を行い、網羅率が1.0になった時点で返信内容を出力する。評価方法は、2つの指標を用いる。1つ目は提案ソフトウェアの出力結果評価フェーズで網羅率が1.0になるまでに入力された改善プロンプトの回数であり、2つ目は提案ソフトウェアが生成した返信内容と、TA が実際に作成した返信内容との意味的な類似性を Sentence BERT とコサイン類似度により数値化し評価を行う。

謝辞

本稿の執筆にあたり、先進情報プロジェクト実習 [IT・3] の TA・SA としてアドバイスをくださった、東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻の大沢恭平さん、東京工科大学コンピュータサイエンス学部の月森陽太さん、坂井萌桜さん、佐藤健斗さん、井出佑さん、山崎雅也さん、手塚雄星さんに御礼申し上げます。

参考文献

[1] Puli, B.: Site Reliability Engineering(SRE) and Observations on SRE Process to make tasks easier (2022).
 [2] Vijayakumar, S. and Taqa, A.: Site Reliability Engineering (SRE), Vol. 2, pp. 2583-9993 (2023).
 [3] Nanda, M. S.: SCALING SITE RELIABILITY ENGINEERING: A DATA-DRIVEN APPROACH TO MOD-

ERN SYSTEM RELIABILITY, *INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN ENGINEERING AND TECHNOLOGY*, Vol. 16, pp. 294-308 (2025).
 [4] Burns, B., Grant, B., Oppenheimer, D., Brewer, E. and Wilkes, J.: Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade, *Queue*, Vol. 14, No. 1, p. 70-93 (online), DOI: 10.1145/2898442.2898444 (2016).
 [5] Mondal, S., Pan, R., Kabir, H. M. D., Tian, T. and Dai, H.-N.: Kubernetes in IT administration and serverless computing: An empirical study and research challenges, *The Journal of Supercomputing*, Vol. 78 (online), DOI: 10.1007/s11227-021-03982-3 (2022).
 [6] Carrión, M.: Kubernetes as a Standard Container Orchestrator - A Bibliometric Analysis, *Journal of Grid Computing*, Vol. 20 (online), DOI: 10.1007/s10723-022-09629-8 (2022).
 [7] Carrión, M.: Kubernetes as a Standard Container Orchestrator - A Bibliometric Analysis, *Journal of Grid Computing*, Vol. 20 (online), DOI: 10.1007/s10723-022-09629-8 (2022).
 [8] Sachdeva, S.: Kubernetes and Docker: An Introduction to Container Orchestration and Management, *International Journal of Computer Trends and Technology*, Vol. 71, pp. 57-62 (online), DOI: 10.14445/22312803/IJCTT-V71I8P109 (2023).
 [9] Turin, G., Borgarelli, A., Donetti, S., Damiani, F., Johnsen, E. B. and Tapia Tarifa, S. L.: Predicting resource consumption of Kubernetes container systems using resource models, *Journal of Systems and Software*, Vol. 203, p. 111750 (online), DOI: 10.1016/j.jss.2023.111750 (2023).
 [10] Wang, H. J., Platt, J. C., Chen, Y., Zhang, R. and Wang, Y.-M.: Automatic misconfiguration troubleshooting with peerpressure, *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI'04, USENIX Association, p. 17 (2004).
 [11] Liu, X., Zhang, P., Abhashkumar, A., Chen, J. and Jiang, W.: Automatic Configuration Repair, *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, Association for Computing Machinery, p. 213-220 (2024).
 [12] Ehud Malul, Yair Meidan, D. M. Y. E. A. S.: GenKubeSec: LLM-Based Kubernetes Misconfiguration Detection, Localization, Reasoning, and Remediation (2024).
 [13] Zhang, C., Liu, L., Wang, J., Wang, C., Sun, X., Wang, H. and Cai, M.: PREFER: Prompt Ensemble Learning via Feedback-Reflect-Refine (2023).
 [14] Team, G., Google and David, O.: Gemini: A Family of Highly Capable Multimodal Models, (online), DOI: 10.48550/arXiv.2312.11805 (2023).
 [15] Reimers, N. and Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, pp. 3973-3983 (online), DOI: 10.18653/v1/D19-1410 (2019).
 [16] Rahman, A., Shamim, M. S. I., Bose, D. and Pandita, R.: Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study, *ACM Transactions on Software Engineering and Methodology*, Vol. 32 (online), DOI: 10.1145/3579639 (2023).
 [17] Bufalino, J., Martin-Navarro, J., Francesco, M. and Aura, T.: Inside Job: Defending Kubernetes Clusters Against Network Misconfigurations (2025).