2021年度 卒業論文 2022/01/27 11:34

メールに対してのQ&Aシステムの開発

大阪産業大学 デザイン工学部 情報システム学科
情報教育システム研究室

18H111 吉村一揮

メールに対しての Q&A システムの開発

18H111 吉村一揮

1　はじめに

授業などで疑問がある時教授に質問することが多々ある。昨今、新型コロナウイルス感染症の影響で大学に登校ができないこともあり質問がメールなどで行われることが増えた。直接質問するよりも簡単なので行いやすいのは良い点だと言える。しかし、教授も常にメールを見ているわけではないので質問してからタイムラグが存在する。また教授には数多くの質問が寄せらる。そのため同じ質問に毎回答えるのは非効率である。

2　目的

教授にはメールで数多くの質問が寄せられる。その質問の中にも新しくされた質問 (以下、新規の質問とする) とすでに回答済みの質問 (以下、既存の質問とする) が存在する。教授は既存の質問にはすでに回答を送っている。数多くの質問があるのに既存の質問に対して 2 度も回答を送るのは非効率であり新規の質問対応へのタイムラグにつながる。本研究の目的は、質問文が書かれたメールを抽出し既存の質問であれば既存の回答を送るシステムを開発し、既存の質問に対しての教授自らの返答を無くすことを目指す。

3　メールに対しての Q&A システム

本研究は、メールで送られた質問 (メイルの本文) を抽出しその抽出した質問が元々教授にされていた質問かどうかを形態素解析 [1] を使用し判断する。もし質問自体が既存の質問の場合は質問に既存の回答を送られてたメールに返信する。逆に新規の質問である場合は既存の質問が書かれたテキストファイルに追記され、教授に新規の質問が来たことメールで通知する。一連の動作を自動でシステムが行うことで教授の負担にならないシステムを実現する。本研究ではシステムの一例として、研究室のメーリングリストに送られた質問を対象とした Q&A システムの開発を行う。

4　検証

本システムを導入した際に質問文をどれだけ正確に読み取ることができ既存の質問か新規の質問なのかの判断ができているかの検証を行なった。検証は 3 つを行う。検証 3 つは下記に示す。

1. 研究室で実際に教授宛に送られた質問文を本文に記載したメールを送信してどのような反応があるか検証
2. 新規の質問を記載したメールを送信して新規メールとして判断するのか検証
3. 既存の質問に違う文だが意味は同じ文を送信した場合どのような反応があるか検証

5　まとめ

本研究では、質問文が書かれたメールを抽出し既存の質問であれば既存の回答を送るシステムの開発を行った。その結果、既存の質問に対してのメールでの返信ができるようになり、本研究であった目的である既存の質問に対して 2 度も返答してしまっている問題の解決に成功した。また今後質問者が増えるごとにサンプルは増えるのでより正確な回答を出せるようになるのも期待できる。しかし本システムでは質問文の名詞のみで文面の判断を行っているため質問文の意味が同じでも名詞が違うだけで違う質問文として判断されてしまっている。それでは文面のニュアンスを掴むことはできないので文章に対する意味の分析をシステムに導入することができればより良いシステムになると期待できる。

参考文献

[1] 形態素解析. 最先端自然言語処理ライブラリの最適な選択と有用な利用方法. https://speakerdeck.com/taishii/pycon-jp-2020?slide=92.

目次

1 はじめに 1

2 目的 2

3 メールに対しての Q&A システムの概要 3
- 3.1 本システムの概要 . . . 3
- 3.2 自然言語処理 . . . 4
- 3.3 動作 . . . 6
- 3.4 既存の質問文とその回答 . . . 6
- 3.5 本システムの詳細 . . . 7

4 検証 14
- 4.1 検証方法 . . . 14
- 4.2 検証結果 . . . 16

5 結果と考察 20
- 5.1 検証結果の考察 . . . 20
- 5.2 本研究で開発したシステムの利点と問題点 . . . 21
- 5.3 利点 . . . 21
- 5.4 問題点 . . . 21

6 結論 22
- 6.1 今後の課題 . . . 22

付録 A ソースコード 25
- A.1 指定した subject のメールに指定したラベルを付与するプログラム . . . 25
- A.2 メールに対しての Q&A プログラム . . . 26

1　はじめに

授業などで疑問がある時教授に質問することが多々ある。昨今、新型コロナウイルス感染症の影響で大学に登校ができないこともあり質問がメールなどで行われることが増えた。直接質問するよりも簡単なので行いやすいのは良い点だと言える。しかし、教授も常にメールを見ているわけではないので質問してからタイムラグが存在する。また教授には数多くの質問が寄せらる。そのため同じ質問に毎回答えるのは非効率である。この問題を解決するため、メールでの質問応答システムが必要であると考える。

第 2 章では本研究の目的について述べる。第 3 章では本研究で開発したシステムの概要を述べる。第 4 章では本研究で開発したシステムを用いて実施した検証について述べる。第 5 章では本研究の結果をその考察とともに述べる。第 6 章には研究の成果とともに今後の課題についてまとめる。

2　目的

教授にはメールで数多くの質問が寄せられる。その質問の中にも新しくされた質問 (以下、新規の質問とする) とすでに回答済みの質問 (以下、既存の質問とする) が存在する。教授は既存の質問にはすでに回答を送っている。数多くの質問があるのに既存の質問に対して 2 度も回答を送るのは非効率であり新規の質問対応へのタイムラグにつながる。

本研究の目的は、質問文が書かれたメールを抽出し既存の質問であれば既存の回答を送るシステムを開発し、既存の質問に対しての教授自らの返答を無くすことを目指す。

3　メールに対しての Q&A システムの概要

本研究は、メールで送られた質問 (メールの本文) を抽出しその抽出した質問が元々教授にされていた質問かどうかを自然言語処理を使用し判断する。もし質問自体が既存の質問である場合は質問に既存の回答を送られてたメールに返信する。逆に新規の質問である場合は既存の質問が書かれたテキストファイルに追記され、教授に新規の質問が来たことメールで通知する。一連の動作を自動でシステムが行うことで教授の負担にならないシステムを実現する。本研究ではシステムの一例として、研究室のメーリングリストに送られた質問を対象とした Q&A システムの開発を行う。また、本学では入学時に学生各自に Google アカウントを付与される。そのため全員が Gmail のアドレスを所有している。なので本研究では Gmail を使用したメールのみを対象として研究を進める。

3.1　本システムの概要

本システム全体の概要図を図 1 に示す。主に 4 つの機能を実装した。次項より各機能の解説を行う。

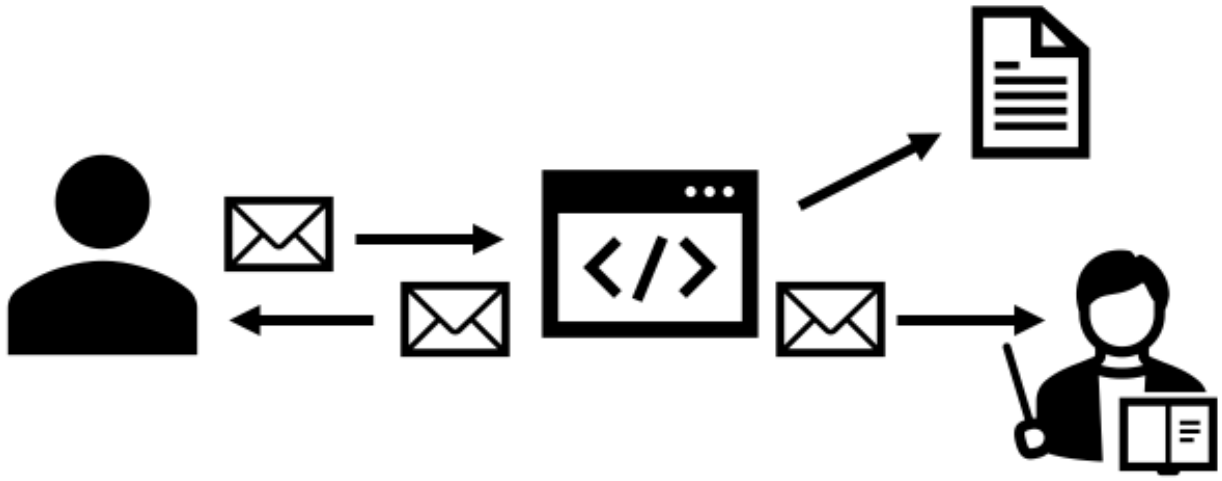

図 1　特定の subject のメールの本文 (質問) を抽出。そこからプログラムで質問自体が既存か新規の質問かを判断する。もし新規の質問である場合、質問文をテキストファイルに追記し教授にメールで通知。また既存の質問である場合、回答をメールで返信する。

1. 特定の subject のメールの本文 (質問) を抽出
2. 質問自体が既存か新規の質問かを判断
3. 質問文をテキストファイルに追記、メール送信による通知
4. 質問文に対しての回答をメールで返信

3.2 自然言語処理

自然言語処理とは、人間の言語 (自然言語) を機械で処理し、内容を抽出することである。または、自然言語が持つ意味をさまざまな方法で解析する処理技術のことである。自然言語の対象としては言葉や文章といったコミュニケーションで使う“話し言葉”から、論文のような“書き言葉”までと定義されている。現在も開発が盛んに行われている開発分野であり、Google 検索エンジンの検索精度向上などにも使用され注目されている分野である。自然言語処理は主に 4 つの工程 (形態素解析、構文解析、意味解析、文脈解析) で処理される。この処理工程のパターンは使用用途によって変わってくる。形態素とは言語学の用語で意味合いとしては“文字で表記された自然言語の文において、意味を持つ最小の言語単位”を指す。形態素解析とはこれを読み取ることで、その言語に固有の“意味を含む情報”を取得し、学習することである。構文解析は別名「係受け解析」とも呼ばれ、形態素解析で取得した単語間の関係性を解析するステップ。これを行うことで、それぞれの単語が互いにどのように作用しているかがわかるようになる。意味解析とは、構造を解析した後の文の正しい意味を解析することである。1 つの単語に内在する複数の意味を、他の単語の関係に応じて適切に指令することで、最も相応しい解釈をコンピューターに学習させる。その過程を経て、候補となった複数の解釈から最適な意味を選択する。文脈解析とは、文章の中に現れる形態素や単語の関係以上の隠された情報の取得することである。この文脈解析と呼ばれるものだが、実はこれを行うシステムは未だに実用化されていない。

本研究では質問文に生じる単語を抽出し、他の質問文で同一単語の割合を見ることで重複している質問を発見するというものである。なので本研究では自然言語処理である形態素解析のみを使用し質問文の意味を持つ最小の言語単位を分解することが適切と考える。

3.2.1 形態素解析ツール

形態素解析にはあらゆる種類のツールが存在し、それぞれに適したものがある。通常よく使われる形態素解析ツール [1] は Mecab,Janome などがある。Mecab は、形態素解析ツールの中でも定番。知名度の高さに加えて、判別精度の高さ、実行速度も早さも特徴として挙げられる。Janome は、Mecab に比べると実行速度は劣る。しかし、Python のみで記述されているため、pip コマンドで簡単に導入できる点がメリットとして挙げられる。実行速度は Mecab は 1 秒に付き 25077 文字で、Janome は 1 秒に付き 698 文字になる。本研究では実行速度を重視したいので Mecab を使用する。例として“すももももももものうちだけど、そんなことはどうでもいいよね？”を Mecab を使って形態素解析を行った。その実行結果を表すターミナルでの画面を図 2 に示す。

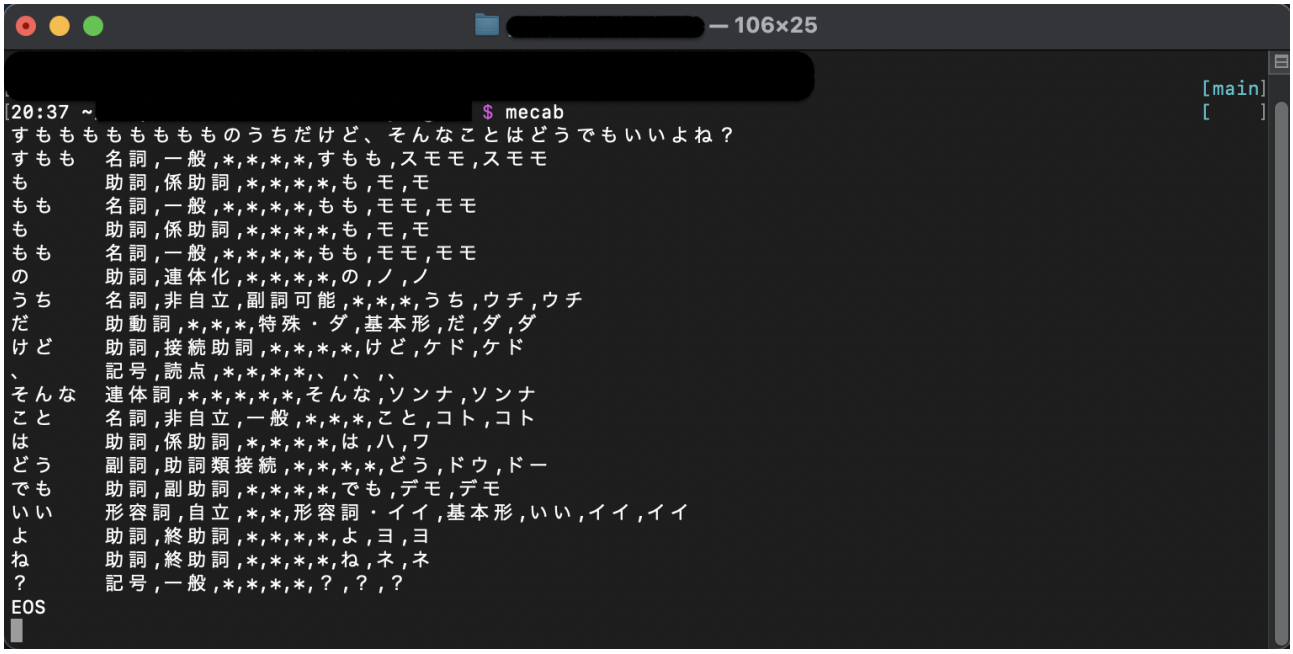

図 2　Mecab を使用し“すももももももものうちだけど、そんなことはどうでもいいよね？”を形態素解析したターミナルの画面。形態素に分割し、それぞれの品詞や変化などを割り出すことが可能。

3.3 動作

以下に本システムの動作を示す。

1. メーリングリストに投げられた特定の subject のメールを特定のラベルを付与。
2. メールの本文、subject、From、messageid を抽出。
3. メール本文を Mecab で形態素解析を行なって名詞のみを抽出。
4. 既存の質問文が一行ずつ書かれた question.txt を Mecab で形態素解析を行なって名詞のみを抽出。
5. 3,4 で抽出した名詞同士をそれぞれ差分して一致率を比較。
6. 新規の質問の場合は question.txt に追記し、教授にメールで通知。
7. 既存の質問の場合は answer.txt に書かれた該当の回答を送られてきた質問メールに返信。

3.4 既存の質問文とその回答

あらかじめ質問文とその回答を用意する。本システムではメールから内容取得などを行うためテキストベースのシステムになる。なのでテキストファイルにデータを格納することでより快適にデータを管理することができるし編集も容易になるので本研究ではテキストファイルを使用する。質問文とその回答をそれぞれ 2 つのテキストファイルで管理する。テキストファイルには 1 行に 1 つずつ質問を格納する。回答も同様である。以下に 2 つのテキストファイルを示す。

1. 既存の質問文を格納するテキストファイル (question.txt)
2. 既存の質問に対しての回答が格納されたテキストファイル (answer.txt)

3.5　本システムの詳細

本システムはプログラム自体が二つ存在しそれぞれ Google Apps Script (以下、GAS とする) [2] と Python で書かれている。

3.5.1　特定 subject のメールにラベル付与

メーリングリストに送られてきたメールの中で未読のこちらで指定した subject を付けたメールのみをラベルをつける。このようにすることで質問メールのみを抽出することが可能。通常特定の subject からメールの内容を取得するためには Google の Gmail API を使用しないといけない。Python などでそれをすると Google などの設定が複雑になってしまう。しかし GAS は Gmail API に容易に使用することが可能であるため今回は GAS でこの処理を行なっている。Gmail のラベルの場所を図 3 に示す。

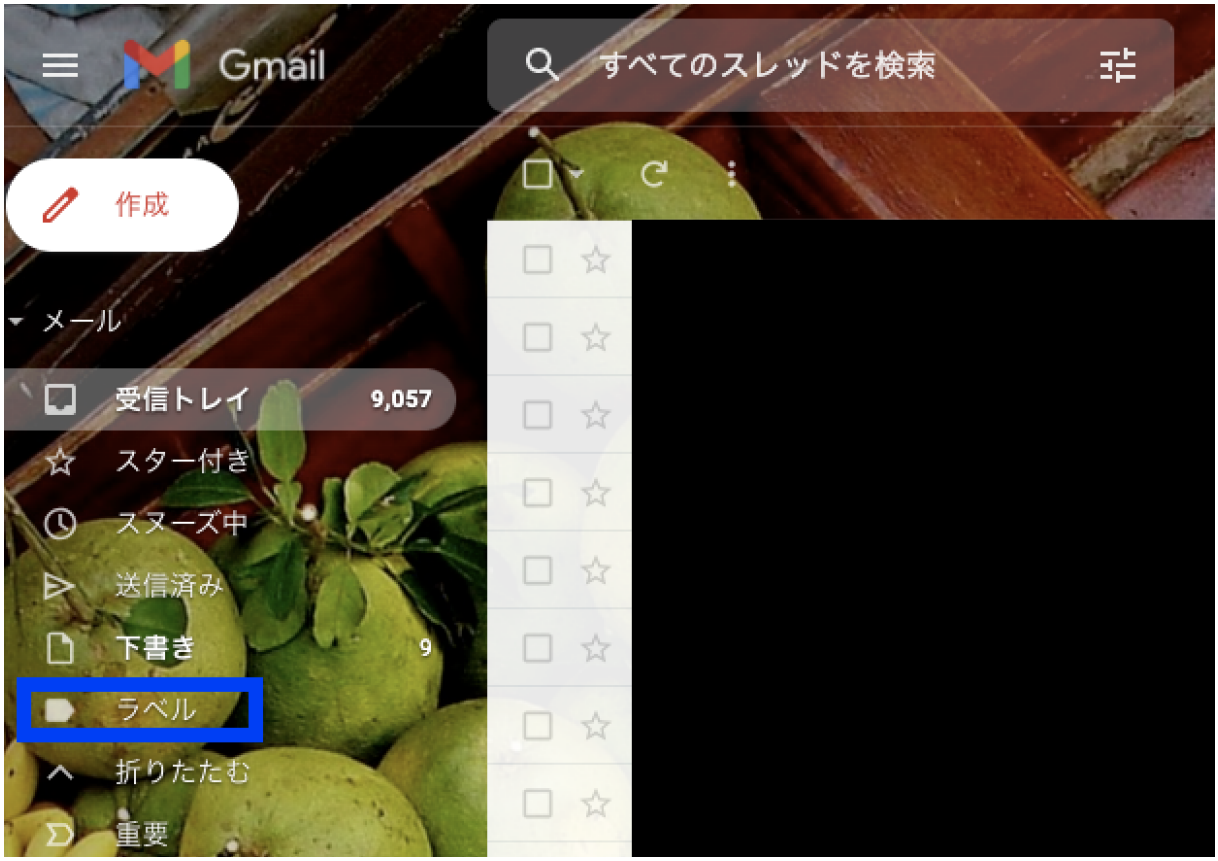

図 3　Gmail 内でのラベルの場所は青枠で囲った場所に存在。指定していた特定の subject のメールはプログラムによってラベル付与される。

3.5.2 メール内容を取得

特定のラベルをつけられているメールの内容を取得する。本研究では Python の imaplib モジュールでメール内容の抽出を行う。取得するものは、メールの本文、subject、メールアドレス、Message-ID の 4 つを取得。Message-ID はメールのユニーク ID のこと。本研究ではメール返信時にメール指定が必要になるのでその際に Message-ID 使用する。

3.5.3 mecab による処理

自然言語処理の Mecab を使ってメールで送られた質問を形態素解析し名詞のみをリスト化する。元々用意している質問文は question.txt に記載している。しかしこれ自体は形態素解析していないのでこちらも形態素解析し名詞のみをリスト化する。本文のみの場合は 1 つのリストのみなのでループさせる必要はない。しかし question.txt に記している質問文は複数存在していると考えられるので記載されている質問文数分のループが必要になる。そのためプログラム自体は別にし、question.txt を処理する方にはテキストファイルの行数文分ループ文を設けて二重リストとして出力する。処理に対しての概要図を図 4 に示す。

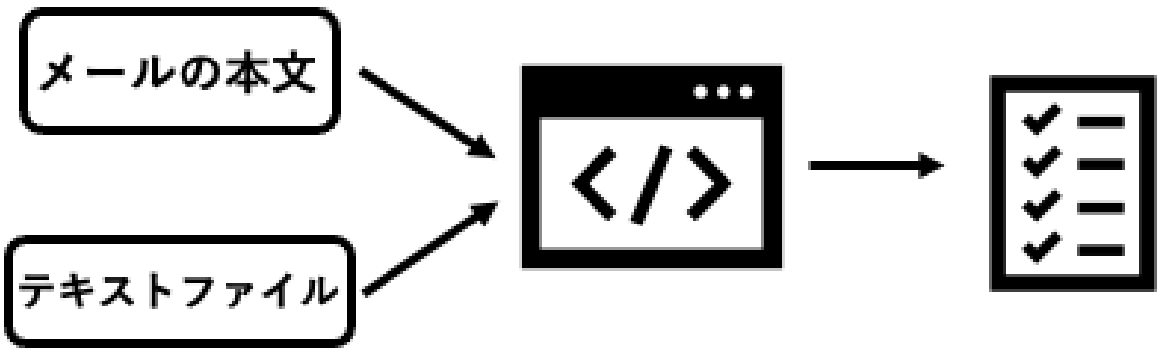

図 4　送られてきたメール本文とテキストファイルを Mecab で形態素解析し名詞のみをリスト化する

3.5.4 リスト同士を差分

それぞれリスト化したものを差分を用いて比較する。Python の difflib ライブラリ内の SequenceMatcher クラスを使用し文字列の類似度を比較する。比較したもの同士の文字列の類似度がこちらで提示した値を超えると既存である質問として判断する。そして既存と判断された質問文の段落番号を出力する。

3.5.5 処理の分岐

もし比較したもの同士の文字列の類似度が低くプログラムが一致する質問文がないと判断する場合、question.txt に追記して教授にメールで新規の質問が来ていることを通知する。逆に比較したもの同士の文字列の類似度が高くプログラムが一致する質問文があると判断する場合、その一致した質問文の行数と同じ行数の answer.txt の答えを抽出する。プログラムの分岐箇所のフローチャート図を図 5 に示す。

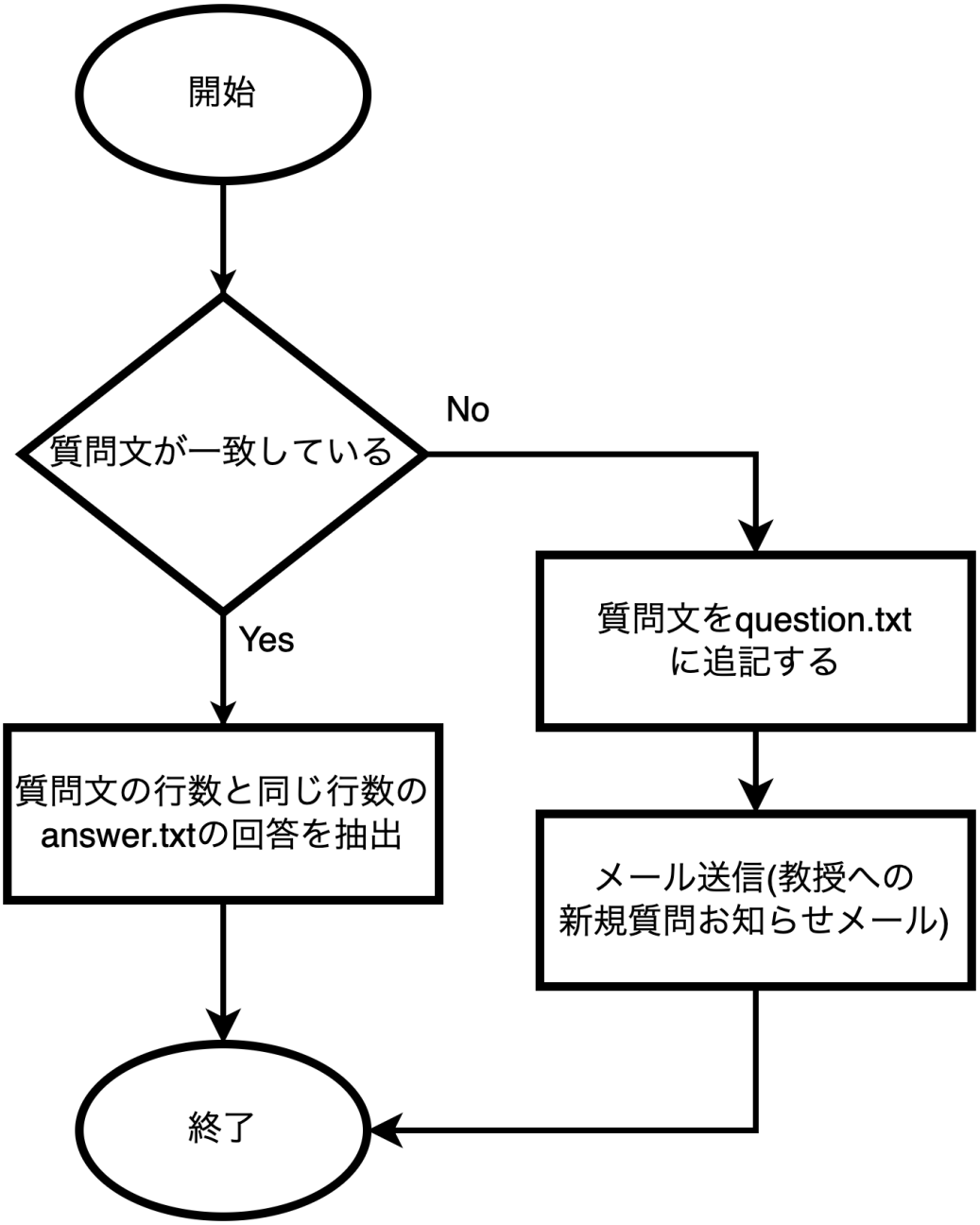

図 5　プログラムの分岐箇所のフローチャート。メールで送られてきた質問文が新規の場合と既存の場合で処理が変化する。もし既存の質問の場合質問文の行数と同じ行数の answer.txt の回答を抽出。逆に新規の質問の場合送られてきた質問文を question.txt に追記し、教授に新規質問がきた通知を送信する。

3.5.6　メールの送信

Python の smtplib モジュール [3] でメールの送信を行う。メールが送信される場合は下記の 2 パターンある。

1. 新規の質問が送られてきたので教授に通知としてメール送信
2. 既存の質問が存在したので質問者からのメールに直接回答を返信

1. は教授にメールを送るだけなのでメール作成して送信する。2. は既存の質問に対しての回答を answer.txt から抽出してメールとして返信する。メールを返信する場合メール指定が必要になる。そのため MessageID でどのメールに返信するかを指定しないといけない。新規の質問が来たときに教授にメールで通知した時のメールの画面を図 6、既存の質問と判断しメールに対して回答を返信している画面を図 7 に示す。

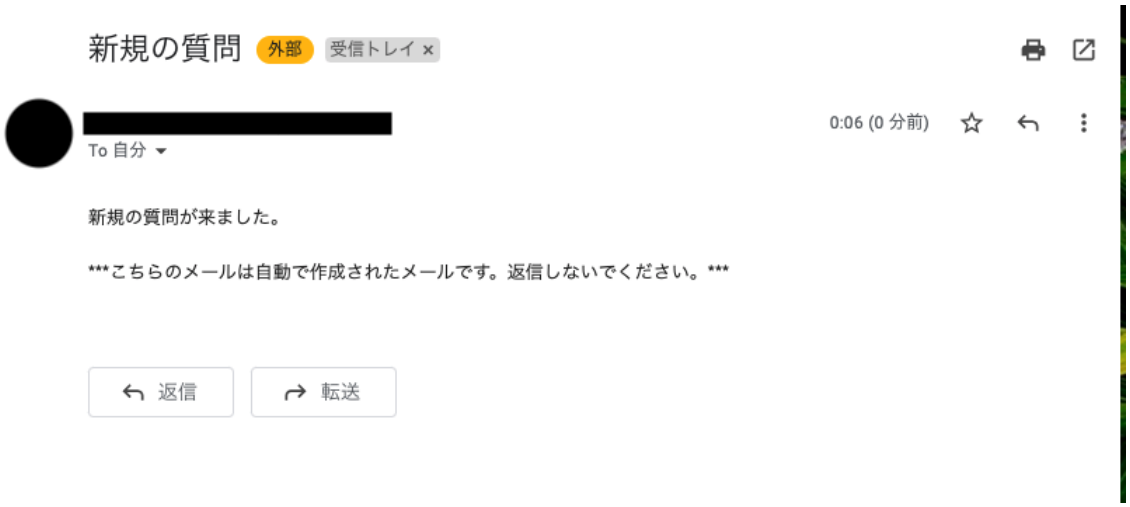

図 6　新規の質問が来たときに教授にメールで通知した時のメールの画面

図 7　既存の質問と判断しメールに対して回答を返信している画面

3.5.7　プログラムの起動

2 つのプログラムをそれぞれ定期的に動かすことで今までのプログラムを一連の動作として動作させる。Python のプログラムの方は cron により定期的に実行。GAS のプログラムは GAS 専用のトリガー設定があるのでそちらで実行する。まず GAS のエディタからメニュー選択の“トリガー”を選択する。その後に“トリガーを追加”を選択する。トリガーの選択場所は図 8 に、トリガー追加の選択場所は図 9 に示す。GAS のトリガーの設定は実行する関数と実行するデプロイはデフォルテの設定。イベントのソースで“時間主導型”を選択。時間主導型を選択することで定期的な時間間隔によるトリガーの指定が可能。そして時間ベースのトリガーで色々な時間的設定が可能。今回はできるだけメール着信からタイムリーにプログラムを実行したいので“分ベース”を選択。時間の間隔の設定は“5 分おき”を選択する。これによってトリガーの設定ができプログラムが定期的に実行される。トリガーの設定画面を図 10 に示す。

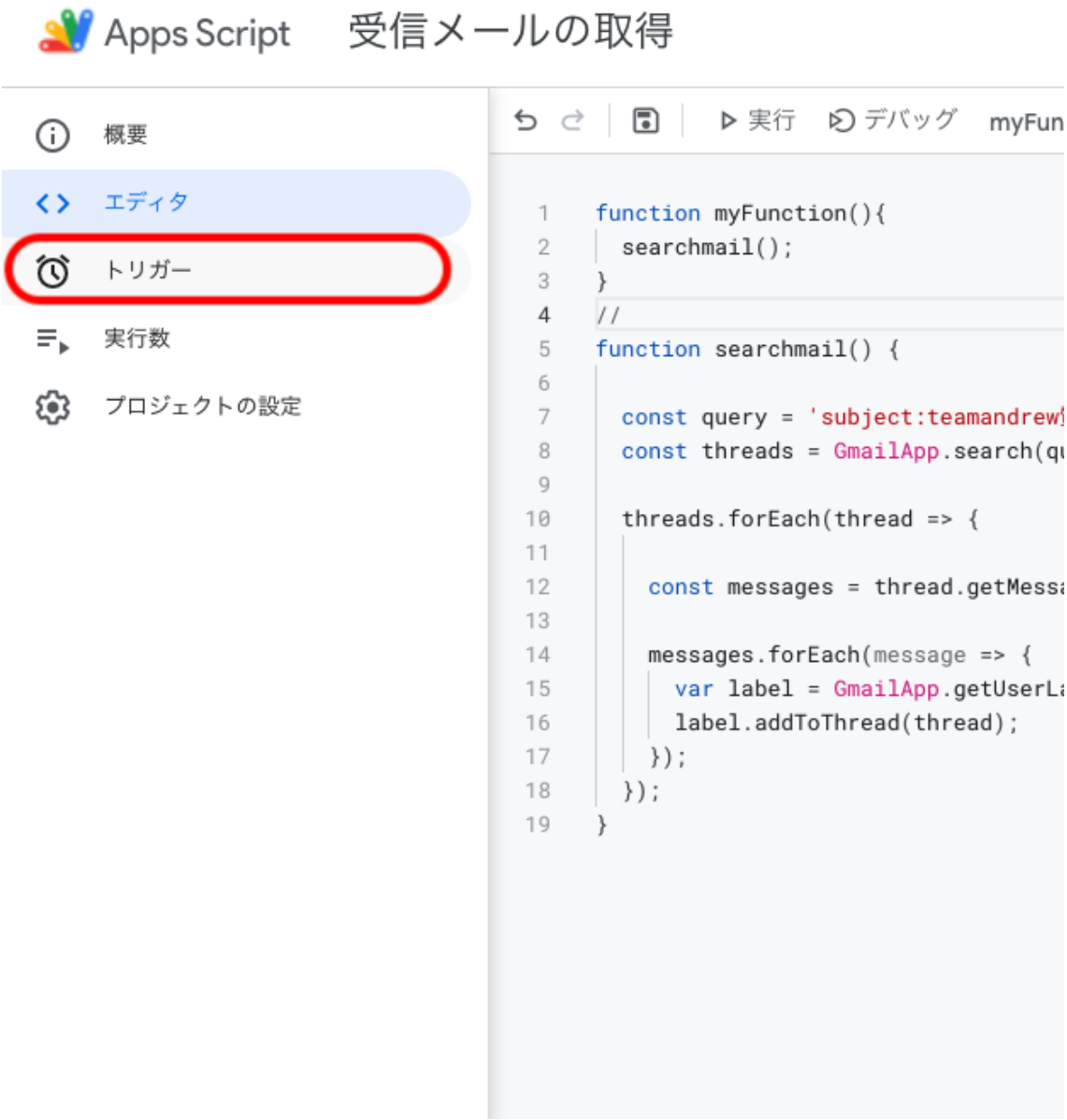

図 8　GAS のエディタからメニュー選択の“トリガー”を選択する。“トリガー”の赤枠で囲ったところを選択する。

図 9　下部にある“トリガーを追加”を選択する。“トリガーを追加”の赤枠で囲ったところを選択する。

図 10　GAS のトリガーの設定画面。今回の研究では実行関数を“myFunction”、デプロイを“Head”、イベントのソースを“時間主導型”、時間ベースのトリガーのタイプを“分ベース”、時間の間隔を“5 分おき”を選択する。

4　検証

本システムを導入した際に質問文をどれだけ正確に読み取ることができ既存の質問か新規の質問なのかの判断ができているかの検証を行なった。

4.1　検証方法

検証は 3 つを行う。検証 3 つは下記に示す。

1. 研究室で実際に教授宛に送られた質問文を本文に記載したメールを送信してどのような反応があるか検証
2. 新規の質問を記載したメールを送信して新規メールとして判断するのか検証
3. 既存の質問に違う文だが意味は同じ文を送信した場合どのような反応があるか検証

前提として研究室で実際に教授宛に送られた質問文とそれに対しての回答をそれぞれ質問文を question.txt、回答を answer.txt にテキストファイルとして書き込みをしているものとする。検証内容はそれぞれ下記に示す。

4.1.1　検証 1

指定の subject と研究室で実際に教授宛に送られた質問文を本文に記載したメールを送信してどのような反応があるか検証する。集めた質問文があった期間は 2021/04/01 から 2021/12/10 まで。質問数は 8 個。以下に研究室で実際に教授宛に送られた質問を示す。

- 研究室はどこですか？
- 研究室行くつもりなんですけど、何時頃ご都合つきますか？
- 教授は今日研究室来れる空いてる時間あります？
- 13 日の土曜日は何時ぐらいまでやりますか？
- 13 時ご都合いかがですか？
- アンケートは再送して頂けましたか？
- 今日の昼前って研究室にいますか？
- 明日の日没までに卒論出せんかったら留年確定って認識であってますか？

4.1.2 検証 2

検証 1 で送信したメールとは別に新規の質問を記載したメールを送信し新規メールと判断し新規メールが来たことを教授にメールするかを検証する。質問文は“あなたは結婚していますか？”としている。送信したメール内容の画像を図 11 に示す。

図 11　検証 1 で送信したメールとは別に新規の質問を送信した際に新規メールと判断するのかを検証する。質問文は“あなたは結婚していますか？”としている。その際のメール送信画面。

4.1.3 検証 3

検証 1、検証 2 で送信したメールとは別に質問文の内容は同じだが質問文自体が違うものを記載したメールを送信してどのような反応があるか検証する。質問文は“研究室はどこにありますか？”、“研究する教室ってどこですか？”、“研究室って何号室ですか？”の 3 つとしている。

4.2　検証結果

4.2.1　検証 1 の検証結果

全ての質問に対して既存質問だと判断し送られてきたメールに質問文に対しての回答を返すことができていた。検証 1 の 1 つ目の質問を実際にメールを送った後にその送られてきたメールに対しての返信をした際の画面を図 12 に示す。

図 12　検証 1 の 1 つ目の質問である“研究室どこですか？”を実際にメールで送った際の質問の回答を記載したメールの返信画面。

4.2.2　検証 2 の検証結果

送った質問メールを新規の質問と判断し教授に新規メールが送られてきたことを知らせる通知メールを送ることができた。また新規の質問文をテキストファイルの question.txt に追記もされていた。教授に新規メールが送られてきたことを知らせる通知メールの画面を図 13、新規の質問文をテキストファイルの question.txt に追記された画面を図 14 を示す。

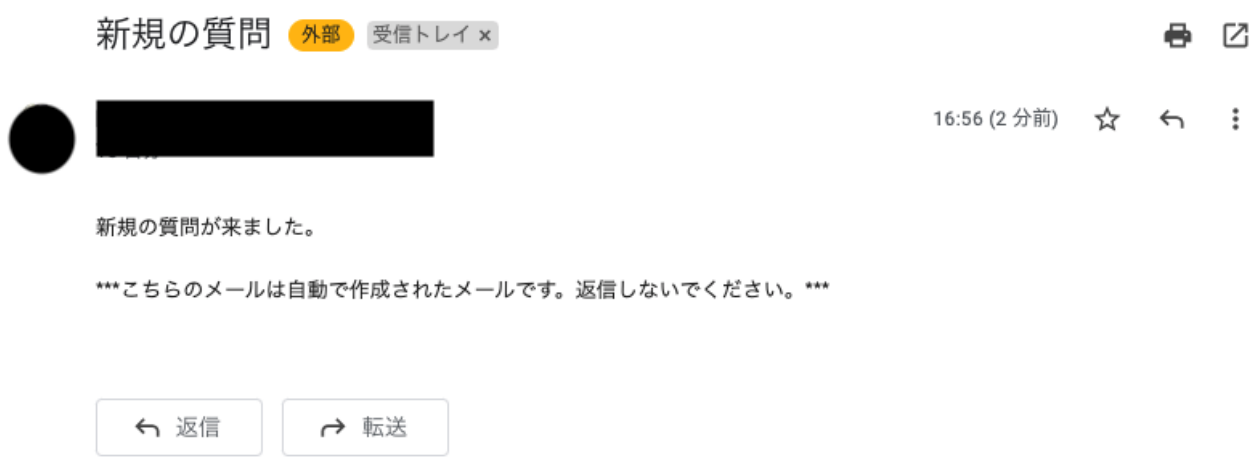

図 13　送った質問メールを新規の質問と判断し教授に新規メールが送られてきたことを知らせる通知メールの画面。

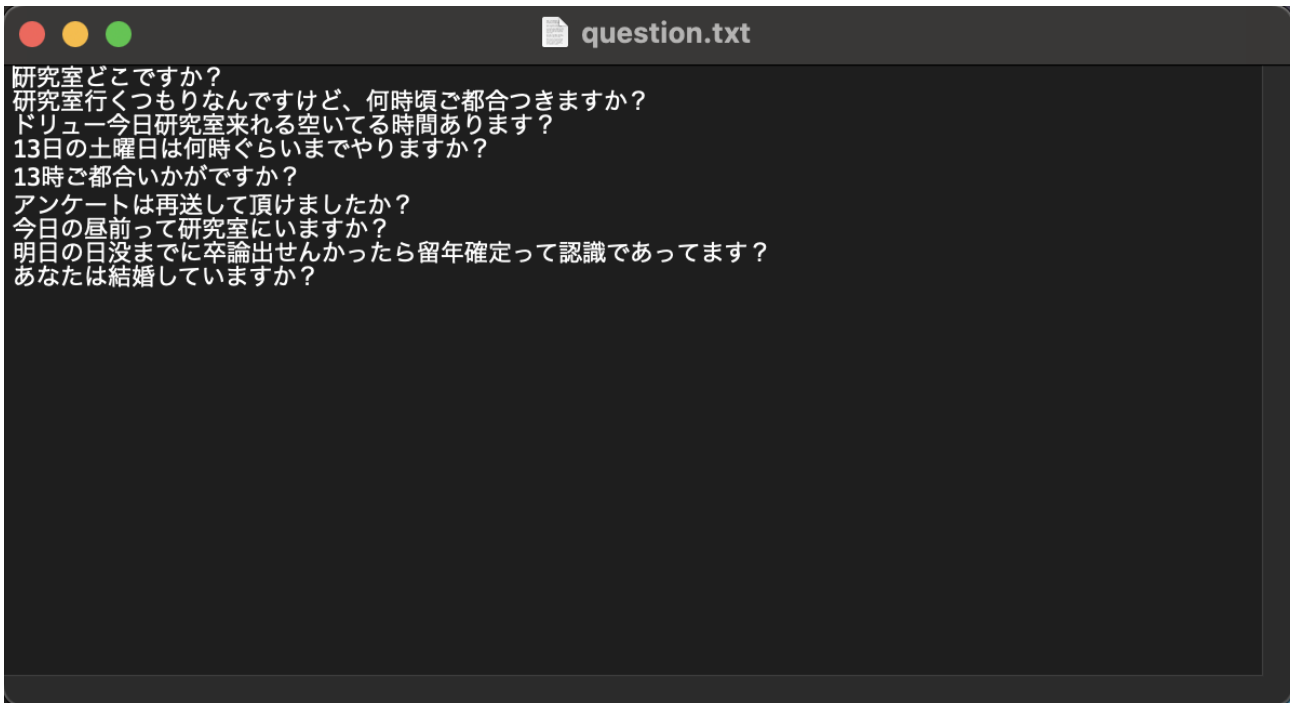

図 14　新規の質問文をテキストファイルの question.txt に追記された画面。

4.2.3　検証 3 の検証結果

1 つ目の質問に対しては既存の質問と判断し既存の質問に対しての回答を送信することができた。2 つ目と 3 つ目の質問に対しては新規の質問と判断され新規質問の通知メールを送信されてしまった。1 つ目の質問に対しての回答の返信メールの画面が図 15、2 つ目と 3 つ目の質問は新規の質問と判断されたため教授への通知メールが送られた画面を図 16、2 つ目、3 つ目の質問を新規の質問と判断し質問文が追記されているテキストファイルの question.txt の画面を図 17 に示す。

図 15　1 つ目の質問を既存の質問と判断し質問の回答を記載したメールの返信画面。

図 16　2 つ目、3 つ目の質問を新規の質問と判断し教授に新規メールが送られてきたことを知らせる通知メールの画面。

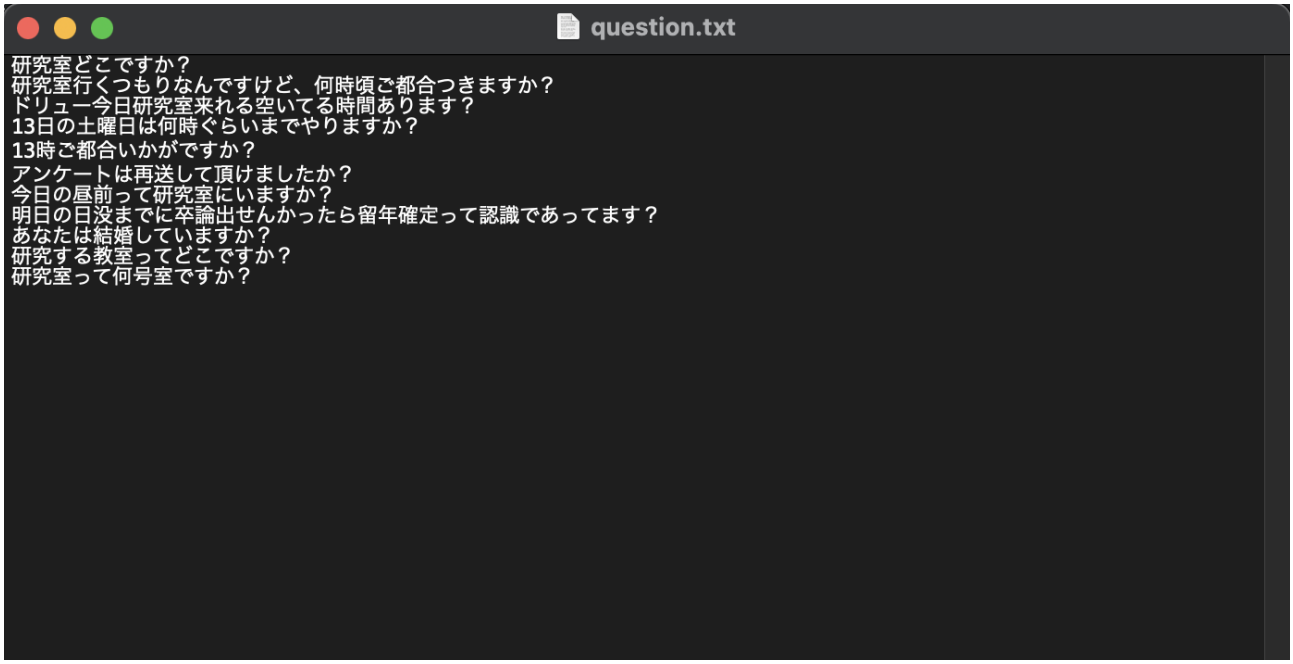

図 17　2 つ目、3 つ目の質問を新規の質問と判断し質問文が追記されているテキストファイルの question.txt の画面。

5　結果と考察

この章では、本研究で開発したメールに対しての Q&A システム、および検証結果に関する考察を行う。

5.1　検証結果の考察

検証結果よりシステム側で既存の質問と新規の質問をそれぞれ判断してそれぞれのプログラム処理をしていたことが分かった。しかし検証 3 で分かったように意味が同じでも質問文の名詞やその名詞の数によって既存の質問か新規の質問かの判断が変わる。また質問文にある名詞が一致しないとどんなに質問が似ていて意味が同じでも既存の質問とは判断されず新規の質問として判断された。検証 3 で送った質問文それぞれの Mecab の出力結果を確認する。“研究室って何号室ですか?”では“何号室”の部分がそれぞれ分解されて 1 つずつが名詞として判断されている。そのため質問文の名詞数が増えているため一致率が下がってしまい新規の質問であると判断されてしまったと考えられる。既存の質問として存在していた質問の“研究室はどこですか?”の Mecab の出力結果を図 18、検証 3 でメールで送信した質問の“研究室って何号室ですか？”の Mecab の出力結果をそれぞれ図 19 に示す。

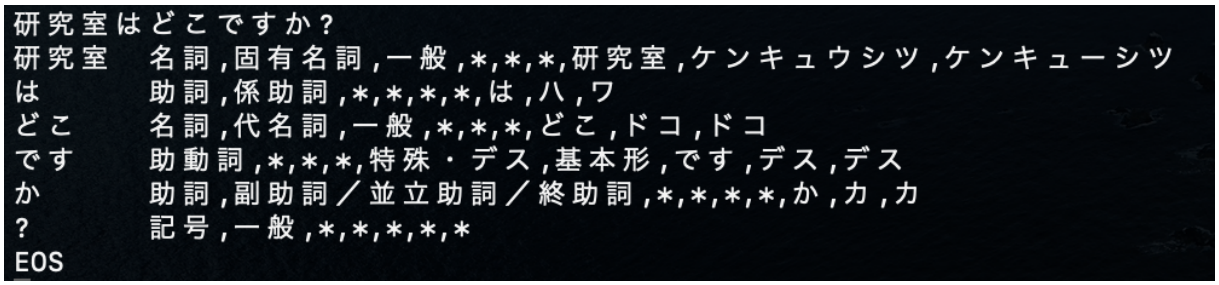

図 18　既存の質問として存在していた質問の“研究室はどこですか?”の Mecab の出力結果のターミナル画面。

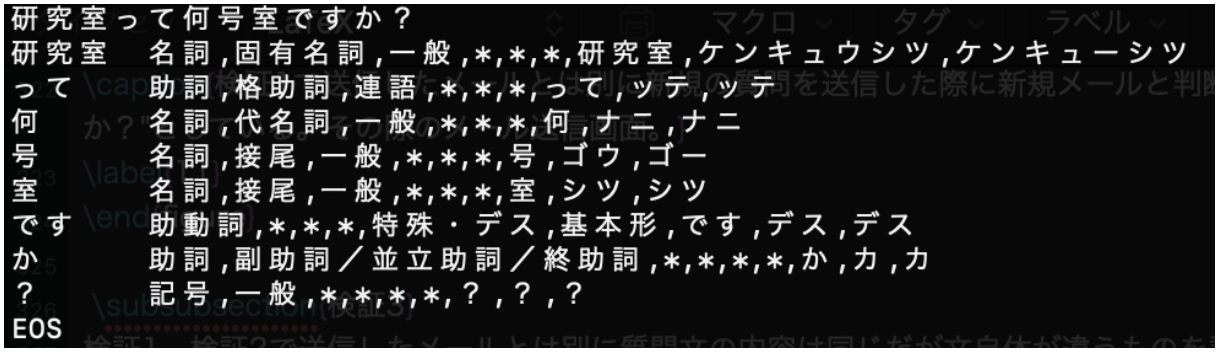

図 19　検証 3 でメールで送信した質問の“研究室って何号室ですか？”の Mecab の出力結果のターミナル画面。

5.2 本研究で開発したシステムの利点と問題点

本研究では質問文が書かれたメールを抽出し既存の質問であれば既存の回答を送るシステムを開発し、既存の質問に対して 2 度も返答してしまっている問題を解決するための手法を提案した。次項で本研究で開発したシステムの考察を示す。

5.3 利点

5.3.1 質問に対して自動で返信する

自動でメールで送られた質問より既存の質問かを判断し既存の質問の場合はそのメールに対して質問の回答を返信することが可能。既存の質問に対して返信を 2 度行わなくて良いのは利点であると考えられる。

5.3.2 データベースがテキストファイル

テキストファイルが質問文とその回答のデータベースになるので直感的に編集ができ処理速度も高速である。

5.3.3 一回のメールに対して複数の質問に対応

一回のメールに対して複数の質問に対応が可能。メール本文の“？”を質問文と判断している。

5.4 問題点

5.4.1 Gmail アカウントの必要性

大阪産業大学では入学時に自身の Gmail アカウントを付与される。そのため本研究では Gmail アカウントを取得していることを大前提としている。しかし Gmail アカウントがないとこの本システムを使うことができない。

5.4.2 プログラムが複数存在

本システムではプログラムが 2 つある。それぞれ違う言語で記述しており別々のトリガーを用いて起動している。そのため一貫性がないため万が一エラーなどで修正が必要な場合に修正箇所の発見が遅れてしまう原因にもなる。

6　結論

本研究では、質問文が書かれたメールを抽出し既存の質問であれば既存の回答を送るシステムの開発を行った。その結果、既存の質問に対してのメールでの返信ができるようになり、本研究であった目的である既存の質問に対して 2 度も返答してしまっている問題の解決に成功した。また今後質問者が増えるごとにサンプルは増えるのでより正確な回答を出せるようになるのも期待できる。しかし本システムでは質問文の名詞のみで文面の判断を行っているため検証でも分かるように質問文の意味が同じでも名詞が違うだけで違う質問文として判断されてしまっている。それでは文面のニュアンスを掴むことはできないので文章に対する意味の分析をシステムに導入することができればより良いシステムになると期待している。

6.1　今後の課題

今後の課題は、質問文の文面を理解することである。上記の内容から以下の課題が考えられる。

6.1.1　文章に対する意味の分析

本システムでは名詞の一致率に対してプログラムの処理をしているがそれだけでは文章のニュアンスを掴むことは難しい。文章のニュアンスを掴むために自然言語処理の形態素解析だけでなく構文解析、意味解析、文脈解析まで使用し質問に対しての意味解析を行う必要があると考える。

謝辞

本研究を進めていく上で、担当教員の大垣 斉准教授からご指導およびご協力を頂きました。また、ご協力頂いた情報教育システム研究室所属の学生および卒業生の方々に深く感謝いたします。

参考文献

[1] 形態素解析. 最先端自然言語処理ライブラリの最適な選択と有用な利用方法. https://speakerdeck.com/taishii/pycon-jp-2020?slide=92.

[2] GoogleAppsScript. Google apps script(gas) とは？ https://anagrams.jp/blog/google-apps-script/.

[3] smtplib. Python でメール送信. https://www.python.ambitious-engineer.com/archives/2034.

付録 A　ソースコード

A.1　指定した subject のメールに指定したラベルを付与するプログラム

Listing 1　指定した subject のメールに指定したラベルを付与するプログラム (searchmail.gs)

```
function myFunction(){
  searchmail();
}指定した
//のメールに指定したラベルを付与するプログラムsubject
function searchmail() {

  const query = 'subject指定した:subject is:unread'
  const threads = GmailApp.search(query);

  threads.forEach(thread => {

    const messages = thread.getMessages();

    messages.forEach(message => {
      var label = GmailApp.getUserLabelByName("指定したラベル");
      label.addToThread(thread);
    });
  });
}
```

A.2 メールに対しての Q&A プログラム

Listing 2 メールに対しての Q&A プログラム (test.py)

```
# -*- coding: utf-8 -*-
from email import message
import imaplib
import re
import email
import base64
import smtplib,ssl
from email.mime.text import MIMEText
from email.header import decode_header, make_header
import MeCab
from difflib import SequenceMatcher

t=MeCab.Tagger('-Ochasen -d /usr/local/lib/mecab/dic/mecab-ipadic-neologd')
ADDRESS = "メールが送られるメールアドレス"
PASSWORD = "アプリパスワード"

def main():
    answer=[]
    r=searchmail()
    for i in r[1]:
        list1 = mecab1(i)
        list2 = mecab()
        l=diff(list1,list2)
        a=bunki(l,r)
        answer.append(a)
    a = '\n'.join(map(str,answer))
    mailsend(a,r)

#メールの内容取得
def searchmail():
    imapobj = imaplib.IMAP4_SSL("imap.gmail.com", '993')
    imapobj.login(ADDRESS, PASSWORD)

    #ラベル下のメールを取得work
    imapobj.select('question')
    typ, data = imapobj.search(None, 'UNSEEN')
```

```
    #取得したメールから表題と本文を出力
    for num in data[0].split():
        typ, data = imapobj.fetch(num, '(RFC822)')

        #メール内容タイトル、本文()
        mail = email.message_from_string(data[0][1].decode('utf-8'))

        #表題出力
        subject = str(make_header(decode_header(mail["Subject"])))

        #本文出力
        b = base64.urlsafe_b64decode(mail.get_payload().encode('ASCII')).decode('utf-8')
        t=b.split('\r')
        body=[]
        for i in t:
            if "?" in i or "?" in i:
                body.append(i.replace('\n',''))

        ad=email.header.decode_header(mail.get('From'))
        add=ad[1][0].decode('utf-8')
        address=re.sub(r"[<>␣]","",add)
        messageid = re.sub(r"[<>␣]","",(mail.get('Message-ID').encode('ASCII')).decode('ut

        return subject,body,address,messageid

    imapobj.close()

#のプログラムmecab送られてきた質問文用()
def mecab1(text):
    res = t.parseToNode(text)
    meishi=[]
    while res:
        arr = res.feature.split(",")
        if (arr[0] == "名詞"):
            #名詞を取り出す
            meishi.append(arr[6])
        res = res.next
    return meishi
```

```
#のプログラムmecab既存の質問文用()
def mecab():
    with open('question.txt', mode='rt', encoding='utf-8') as f:
        read_data = list(f)
        l = []
        for i in read_data:
            res = t.parseToNode(i)
            meisidousi=[]
            while res:
                arr = res.feature.split(",")
                if (arr[0] == "名詞"):
                    #名詞を取り出す
                    meisidousi.append(arr[6])

                res = res.next
            l.append(meisidousi)
    return l

#文字列の差分を比較
def diff(list1,list2):
    list=[]
    score=0
    count=0
    for i in list2:
        s = SequenceMatcher(None, list1, i)
        #print類似度:, ('{0}%{1} {2}'.format(round(s.ratio()*100,1), list1,i))
        if ((round(s.ratio()*100,1))>=50.0):
            if(score<(round(s.ratio()*100,1))):
                score=round(s.ratio()*100,1)
                list.append(count)
        count+=1
    return list

#プログラムの分岐ポイント
def bunki(l,r):
    if (not l):
        #空の場合は教授に通知と来た質問をquestion.に追加txt
        with open('question.txt', mode='a', encoding='utf-8') as f:
            f.write('\n'+r[1])
            mailsend1(r)
```

```
    else:
        #有の場合は質問文に対する回答をanswer,から抽出txt
        with open('answer.txt', mode='rt', encoding='utf-8') as f:
            read_data = list(f)
            count=0
            for i in read_data:
                if (count in 1):
                    answer=(f'{i}')
                count+=1
        return answer

#メール送信のプログラム回答返信用()
def mailsend(a,r):
    smtpobj = smtplib.SMTP_SSL('smtp.gmail.com',465,context=ssl.create_default_context())
    smtpobj.login(ADDRESS, PASSWORD)

    bodyText = a
    fromAddress = ADDRESS
    toAddress = r[2]
    toSubject = str(f'Re: {r[0]}')
    messageid = r[3]

    # メール作成
    msg = MIMEText(bodyText,"plain",'utf-8')
    msg['From'] = fromAddress
    msg['To'] = toAddress
    msg['Subject'] = toSubject
    msg['In-Reply-To'] = messageid

    # 作成したメールを送信
    smtpobj.send_message(msg)
    smtpobj.close()

#メール送信のプログラム通知送信用()
def  mailsend1(r):
    smtpobj = smtplib.SMTP_SSL('smtp.gmail.com',465,context=ssl.create_default_context())
    smtpobj.login(ADDRESS, PASSWORD)

    bodyText = """新規の質問が来ました。こちらのメールは自動で作成されたメールです。返信しないで
ください。
```

```
******
"""
    fromAddress = ADDRESS
    toAddress = '教授のメールアドレス'
    toSubject = '新規の質問'

    # メール作成
    msg = MIMEText(bodyText,"plain",'utf-8')
    msg['From'] = fromAddress
    msg['To'] = toAddress
    msg['Subject'] = toSubject

    # 作成したメールを送信
    smtpobj.send_message(msg)
    smtpobj.close()

if __name__ == "__main__":
    main()
```