06　産技短 LINE アカウントへのチャット Bot 機能の実装

戸田　結子，平野　健斗

指導教員　石舘　勝好

1.　はじめに

コロナ禍により，産技短を知る機会が減ったことを踏まえ昨年度の卒業研究として「産技短公式 LINE アカウント高校生向け Q&A 機能の実装」が行われた．この研究では利用者からの質問に対して自動で応答する仕組みや，Q&A データベースの作成が行われたが利用者が質問文を入力するのではなく階層メニューからの選択式であり，階層メニューを追っていかなければ知りたいことにたどり着けないというデメリットがあった．そこで研究を引き継ぎ，デメリットを解消し，公式 LINE の実用化を目標にこの研究に取り組んだ．

図 1　昨年度作成された LINE の画面

2.　研究概要

2.1　チャット Bot 機能の実装

昨年のシステムは階層メニューが用意されており，メニューを選択していくと最終的に回答を返信するというものであり，ユーザから直接入力した質問文については対応していなかった．そこで現在の産技短公式アカウントにチャット Bot 機能を追加した．

チャット Bot はユーザから送られた質問文に最も類似している質問文を CSV ファイルから探し、「(ユーザが送ったものに近い質問文)の回答です」「(回答)」を返す．近い質問内容が見つからない場合は学校のホームページを参照するよう促す．

また，「20 字を超えた文を送った場合」「文章以外（スタンプ）を送った場合」のいずれかの場合はもう一度質問を送るように促す．動作の様子について図 2 に示す．

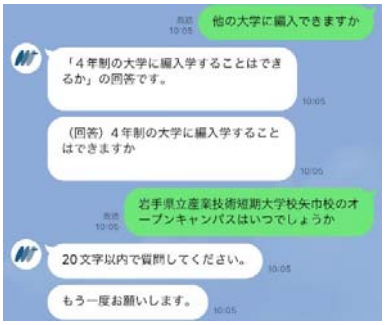

図 2　チャット Bot 機能の動作

また，LINE アカウントを管理する LINEDevelopers においてユーザが特定の言葉を送ると特定の返信をする機能を設定している．この機能では「質問」「矢巾校までのアクセス」「水沢校までのアクセス」「アンケート」の 4 つの言葉への返信文を設定した．

2.2　リッチメニューのデザイン，機能の変更

昨年度のシステムのリッチメニューに機能の追加や削除を行い，4 つの機能に変更した．変更後のリッチメニューを図 3 に示す．

図 3　変更後のリッチメニュー

これらのリッチメニューのデザインの変更は LINE Developers で行った．4 つの機能については以下の通りである．

・産技短のホームページを開く
・質問の例，注意事項を送信する
・資料請求のページを開く
・矢巾校または水沢校のアクセス情報を表示する

図 4 は「アクセス」を選択した際の動作である．

図 4　「アクセス」を選択した際の動作

3.　システム構成

3.1　全体構成

この LINE システムを作成するうえで，アカウントを通じてユーザとの双方向コミュニケーションを実現する「Messaging API」を利用する．これは LINE を介してユーザのメッセージが Webhook に送信される仕組みである．研究では，Bot エンジンを実行する処理も Webhook 内に記述する．

Bot エンジンは,Webhook から送られてきた質問文と最も類似する質問文を CSV ファイル内から見つける仕組みである.類似文書を見つける過程で TF-IDF やコサイン類似度という手法や MeCab,WordNet を用いる．

システム構成を図 5 に示す．

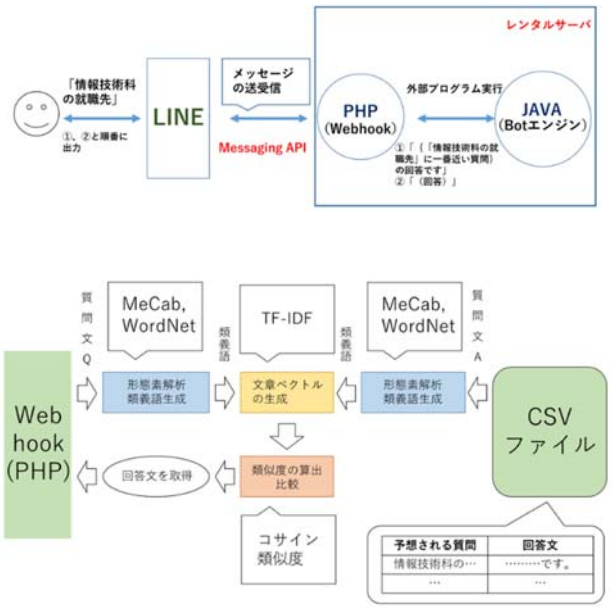

図 5　システム構成図
（上　全体の構成　下　Bot エンジンの構成）

3.2　開発環境

LINE Developers を用いて LINE の様々なシステムを実装するほか，MeCab を用いて形態素解析，Wordnet を用いて類義語の検索を行う．

開発環境は表 1 の通りである．

表 1　開発環境

OS	Windows10，Linux
言語	PHP，Java
サーバ	ConoHa　VPS （メモリ　512MB，CPU　1 コア）

3.3　Webhook と Bot エンジンの連携

LINE と Bot エンジンのやり取りをするために Webhook に Bot エンジンとの連携を記述した．

本研究では PHP から外部プログラムを実行できる関数 exec を用いて Java プログラムである Bot エンジンを呼んでいる．ユーザから受け取った質問文を Bot エンジンに引数として渡し，返り値として回答文を返す．日本語を受け取るには外部プログラムを実行する際にサーバと同じ環境変数 LANG を日本語に設定する必要がある．

3.4　MeCab について

MeCab とは，京都大学情報学研究科―日本電信電話株式会社コミュニケーション科学基礎研究所　共同研究ユニットプロジェクトを通じて開発されたオープンソース形態素解析エンジンである．質問文を形態素解析し単語に分ける際に用いる．

3.5　WordNet について

WordNet とは，世界的に有名な概念辞書．英語 WordNet をもとに日本語 WordNet が構築されている．また，今研究では，日本語 WordNet データベースの JavaAPI である JAWJAW(JAva Wrapper for JApanese Wordnet)を使用し，形態素解析された質問文中の単語の類義語を取得する．

3.6　TF-IDF について

TF-IDF とは, 文書中に含まれる単語の重要度を評価する手法の 1 つであり，TF(単語の出現頻度)と IDF(逆文書頻度)の 2 つの指標に基づいて計算される．

TF(Term Frequency)とは, 単語の出現頻度のことで，各文書においてその単語がどのくらい出現したのかを意味する．TF の求め方を図 6 に示す．

$$\mathrm{TF} = \frac{\text{文書A における単語X の出現頻度}}{\text{文書A における全単語の出現頻度の和}}$$

図 6　TF の求め方

IDF(Inverse Document Frequency)とは、逆文書頻度と呼ばれており、ある単語が全文書中のどれだけの文書で出現したかの逆数を意味する。そのため、IDF は一種の一般語フィルタとして働き、多くの文書に出現する語(一般的な語)の重要度を下げ、特定の文書にしか出現しない単語の重要度を上げる．IDF の求め方を図 7 に示す．

$$\mathrm{IDF} = \log\left(\frac{\text{全文書数}}{\text{単語X を含む文書数}}\right)$$

図 7　IDF の求め方

TF-IDF は TF と IDF を掛け合わせることで求めることができる．TF-IDF の求め方を図 8 に示す．

$$\text{TF-IDF} = \mathrm{TF} * \mathrm{IDF} = \left(\text{単語の出現頻度}\right) * \left(\text{各単語のレア度}\right)$$

図 8　TF-IDF の求め方

これらのことから，出現頻度が高く，レア度が高い(その単語が含まれる文書数が少ない)ほどその単語の重要度は高くなり，逆に出現頻度が低いまたは多くの文書に出現する単語は重要度が低くなる．

3.7　コサイン類似度について

コサイン類似度とは，2 つのベクトルがどのくらい似ているかを表す尺度である. 具体的には(ベクトル空間における）2 つのベクトルがなす角のコサイン値のことである．この値が 1 に近ければ類似しており，0 に近ければ似ていないことになる．コサイン類似度の求め方を図 9 に示す．

$$\cos(\vec{p},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\cdot\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

図 9　コサイン類似度の求め方

3.8　Bot エンジンの構成

Webhook から送られてくる質問文の文字列と CSV ファイル内の質問文の文字列の類似度を求め，最も類似している質問文に対応した回答文を返すシステムである．

Bot エンジンの動作は以下の通りである．

・Webhook から送られてきた質問文と CSV ファイルの中にある質問文をそれぞれ MeCab によって形態素解析し，WordNet で類義語を生成する．

・生成された類義語を用い，TF-IDF とコサイン類似度で二つの質問文の類似度を求める．その操作を CSV ファイル内の全ての質問文に対し行う．

・求めた類似度をそれぞれ比較し，一番類似度が高い質問文に対応する回答を返す．

4.　システムの評価

4.1　ユーザインタフェース

初めて使う人が混乱しないようにメニュー（機能・デザイン）や返信内容を作成した．

リッチメニューのイラスト・画像はそれぞれどのような機能があるかわかるようなものに変更した．視認性に関しては第三者による確認を得ていないが文字が目に入ってくるような色を組み合わせた．また，返信する際に自分の質問したものと一致する質問内容がわかるようにした．

4.2　システムの応答時間

LINE に入力してから回答が最短 5 秒から最長 10 秒程で返ってくる．少し間が開いてしまうので，事前に時間がかかるという旨のメッセージを送ることとしている（図 10）．Windows の開発環境では応答時間が 3 秒ほどで安定しているので，レンタルサーバの性能をよくすれば応答時間も短縮されると思われる．

しかし，複数台の端末から同時に質問文が送られてくると処理に倍以上の時間がかかりエラーとなる．これは Messaging API の仕様により，約 30 秒以内に処理が終わらないと回答を返せないためである．

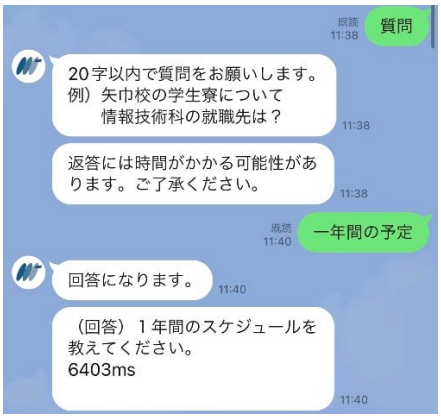

図 10　回答と実行時間

4.3　回答の正確性

送られてくる質問に対して，概ね希望通りの回答を返すことができる．質問文だけでなく単語で送られてきた場合にも対応することが可能である．しかし，用意されていない回答の質問がされたとき，他の回答の単語に引きずられて全く違う回答が返される場合がある．

テストケース（一部）を図 11 に示す．

・「オープンキャンパスはいつありますか」
→「オープンキャンパスはいつ」の回答
・「オープンキャンパス」→「オープンキャンパスについて」の回答
・「1 年間の予定」→「1 年間のスケジュールについて」の回答
・「学校見学はありますか」→「自動車学校は通えるか」の回答

図 11　テストケース（一部）

5.　終わりに

本研究では，目標としていた実用化に向けて産技短公式アカウントにチャット Bot 機能を追加することが出来た．リッチメニューの画像をオリジナルなものに変更することや新しい機能の追加も完了した．課題だったシステムの応答時間も最短で 5 秒までに短縮できた．このことから，はじめに挙げたデメリットを解消できたといえる．

しかし，複数端末で同時にメッセージを送信されたときに起こるエラーを解消することは出来なかった．これはレンタルサーバの性能をあげるかエラーが生じたときにメッセージを再送信するプログラムを実装すれば実用化が可能になると考える．

6.　参考文献

1）Messaging API リファレンス：
https://developers.line.biz/ja/reference/messaging-api/
2）MeCab:https://taku910.github.io/mecab/
3）日本語 WordNet :
http://compling.hss.ntu.edu.sg/wnja/index.ja.html
4）tf-idf – Wikipedia :
https://ja.wikipedia.org/wiki/Tf-idf
5）tf-idf についてざっくりまとめ_理論編 | DevelopersIO :
https://dev.classmethod.jp/articles/yoshim_2017ad_tfidf_1-2/
6）コサイン類似度 :
https://www.cse.kyoto-su.ac.jp/~g0846020/keywords/cosinSimilarity.html