スタッフ離職防止のためのLINE Botアプリの仕様

製品の概要：

LINEを使った一般の会社組織のスタッフのコンディション確認用のBotアプリ。

自動応答・チャットシステムによるスタッフの心身の状態を確認するためのもの。

製品の目的：

職場管理者による介護スタッフへのヒアリングの自動・効率化、

および、介護スタッフの離職を事前に察知・予見して防止すること。

使用する技術・製品：

1. line-bot-sdk(クライアント端末にスマートフォンを想定しているので)。
2. Python(Botの中核であるAIエンジンを組み込む必要があるので)。
3. Flask(Python単体ではアプリケーションの短期開発は難しいので)。
4. (本番・運用環境としては、)GCP(もしくはAWS)。

※(開発＆テスト環境としては、)Heroku。

⑤　使用データベースについては、現在検討中。

　　※発話・メッセージの登録・保存には、No SQL系DBを、

　　ユーザー情報の登録・保存には、RDB(＝SQL)を使います。

⑥　Janome(発話・メッセージの形態素解析をする必要があるので)。

関連書面(＝仕様書)：

1. アプリ全体のクラス図＆シーケンス図等のUML

(実装・開発スピードを優先する場合、用意する必要なし)。

1. 該当アプリを開発・実現する上で求められる、

エコシステム(＝外部連携システムとの関係)について書かれたもの(最低限必要)。

③ その他の文面(ワークフローとその補足事項)。

その他・備考等：

1. 施設利用者の森下が、単独で該当アプリを実装・開発します(＝ソロ開発)。
2. 実装・開発の途上で不明な点があったら、阿部先生に意見・助言を求めます。

該当LINE Botアプリのエコシステム

(＝使用する技術・製品同士の関係＆全体像)

Python AIエンジン　 　line-bot-sdk

RDB and No SQL DB

Janome Flask

GCP or AWS

※各要素間に引かれた直線は、制御の流れを表します。

※縦方向が垂直統合、横方向が水平統合を表します。

上記、４つの技術・製品を使って該当アプリを構成します。

AIエンジンが中核であり、これを中心に実装・開発します。

LINE Bot内部の処理の流れ(＝ワークフロー)

1. 「会話の起点・分岐点(＝スウィッチングワード)の検出」

例：「はじめまして」「こんにちは」、「ところで」「では」・・・等。

1. 「スウィッチングワードに応じた会話の流れの生成」

ユーザーの発話の意図を汲み取って、

会話の流れ(＝フロウ, conversation-of-flow)を決定する。

※「フロウ」とは、”会話のパターン”のようなもの。

1. 「②で決定したフロウに沿った実際の会話の展開・実施」

この段階では、できるだけキャッチーな言葉や投げかけを駆使する。

ユーザーの興味や関心を十分に惹きつけられるようにする。

1. 「ユーザーからの応答の速さ・内容を基にしたユーザーの反応・感触の確認」

AIエンジンが会話を自己評価・採点(＝スコアリング)をする。

1. 「スコアリングの結果をもとに、現在の会話を続けるか、中止するかの判断」

ユーザーの反応・感触が良ければ⑥へと進み、悪ければ①に戻る。

1. 「ユーザーが現在抱えている悩みや問題の聞き出し(＝実質的なヒアリング)」

心理分析・診断用エンジンへと実行の切替えをする。

心理学・精神病理学等のモデルに則った質問を実施する。

1. 「ユーザーの心理・精神状態を分析した症状や病質の判定」

この段階でユーザーとの会話を終了(＝クローズ)して⑦へと進む。

1. 「これまでの会話内容や、分析・診断の結果・記録等のセーブ」

ユーザーとの会話の内容や、分析・診断の結果・記録等をデータベースに登録する。

1. 「⑦の判定結果を受けた職場・現場管理者への通知」

あらかじめデータベース(RDB)に登録されている管理者にのみ通知をする。

1. 「制御・処理サイクルの完了・一巡」

一時情報(＝キャッシュ)を破棄し、使用リソースを解放して①に戻る。

前項(＝ワークフロー)に関する補足事項

前項②についての補足：

ユーザーとの会話の中で、既存の学習辞書に含まれない単語や語句を検出したら、

ネット上のオンライン辞書を参照して、その単語や語句を辞書に追加・登録する。

※ボットの参照する辞書は、「学習辞書」と「ネガティヴワード辞書」とに分けられている。

ユーザーからの発話・メッセージを、１次・２次・３次フィルターにかけて処理する。

１次フィルターは、誤字・脱字・表記の揺れを修正してストップワードを除去する。

２次フィルターは、形態素解析をして、品詞や単語単位に分解をする。

３次フィルターは、構文・意味解析をして、その意図を抽出する。

抽出された発話・メッセージの意図から、今後のフロウを生成する。

最終的には、ユーザーを心理分析・診断へと誘導する格好のフロウを生成する。

前項③についての補足：

②で生成されたフロウに適合する発話・メッセージを生成する。

前項④についての補足：

ユーザーの気質・性格的な傾向の分析・判定には、「ビッグファイヴ理論」を適用する。

前項⑥についての補足：

ボットが心理分析・診断モードに移行(＝エンジンの切替え)をしてからは、

参照する辞書の優先順位を「ネガティヴワード辞書」→「学習辞書」とする。

※この段階では、「学習辞書」に登録されている単語や語句への反応パターンを、

「ネガティヴワード辞書」のそれへの反応パターンに上書き(＝オーバーライド)する。

前項⑨についての補足：

データベースに登録される個別ユーザーとの会話の中身や、

分析・診断の結果、記録等を管理者用ツールで閲覧ができる。

前項⑩についての補足：

管理者への通知としては、分析・診断時点での、

ユーザーの心理・精神的危険度(＝リスクレート)と、

精神症状や精神病質についての分析・判定結果を含める。

発話の意図するもの(＝インテント, intent)の分類

表明(現在＆肯定)・・・「～する」等。

表明(現在＆否定)・・・「～しない」等。

表明(現在進行＆肯定)・・・「～している」「～しています」等。

表明(現在進行＆否定)・・・「～していない」「～していません」等。

表明(現在進行＆可能＆肯定)・・・「～できている」等。

表明(現在進行＆可能＆否定)・・・「～できていない」等。

表明(過去＆可能＝完了)・・・「～できた」「～できました」等。

表明(過去＆不可能＝未完了)・・・「～できていない」「～できていません」等。

表明(現在＆可能)・・・「～できます」「～できる(と思う, と思います)」等。

表明(現在＆不可能)・・・「～できない」「～できません」等。

表明(未来＆肯定)・・・「～しよう(と思う)」「～したい(と思う)」。等。

表明(未来＆否定)・・・「～しないと(思う)」「～したくない(と思う)」等。

表明(過去＆肯定)・・・「～した」「～しました」等。

表明(過去＆否定)・・・「～してない」「～していません」「～しませんでした」等

紹介＆説明＆提示・・・「(～は)～です」「((～は)～でした」等。

疑義＆質問・・・「～？」「～か？」「～ですか？」「～でしょうか？」等。

誘導＆勧誘・・・「～しよう(か?)」「～しましょう(か?)」等。※カッコ内は「半疑問」。  
欲求＆欲動・・・「(～を)～したい(な, です)」「(～を)やりたい」等。

制止＆禁止・・・「(～を)するな」「(～を)しないよう(に)」等。

依頼・・・「(～を)して(ください)」等。

命令・・・「(～を)しろ」「(～を)しなさい」等。

挨拶・・・「おはよう(ございます)」「こんにちは」「こんばんは」「やあ」「どうも」等。

称賛・・・「さすが(ですね)！」「素晴らしい！」等。

罵倒・・・「最低！」「バカ！」等。

称賛(半疑問)・・・「(あなたは)天才ですか？」等。

罵倒(半疑問)・・・「(あなたは)バカですか？」等。

その他

呼掛け・・・「おい」「ねぇ」「なぁ」等。

掛合い(コールアンドレスポンス)・・・「海！」等。※「山！」と答えるのが適当。

※さらに、各インテントには、「能動態」と「受動態」との別があります。

発話の意図されるもの(＝コンテント, content)の分類

以下の通り、コンテントは、大きく２種類に分けられます。

「5W1H」＝思考＆事実の優先＝比較的長文。

「紋切り＆言切り」＝感情＆態度の優先＝比較的短文。

「5W1H」は、その名の通り、「Who」「What」「When」「Where」「How」で、

相手に対して、自分の思考や、把握している事実を優先して表して、これを伝えます。

「紋切り＆言切り」は、ワンワードであったり、セリフのようなもので、

相手に対して、自分の感情や態度を優先して表して、これを伝えます。

インテントの検出・抽出方法

１. 発話・メッセージを１次フィルターによって、書式整形します。

２. 発話・メッセージを２次フィルターによって、形態素解析します。

３. 発話・メッセージの末尾の部分からインテントを検出・抽出します。

４. 発話・メッセージの中からインテントを除去して、コンテントのみの状態とします。

コンテントの検出・抽出方法

１. インテントを検出・抽出・除去したあとの、

発話・メッセージ(＝コンテントのみ)を形態素解析します。

２. 発話・メッセージを、その主体と、

主体(≒主語)の「動作」「状態」「関係」に分けて、マップに格納します。

※主体の動作・状態・関係に係っている、

特定の修飾(＝副詞・形容詞等)も、それぞれの一部としてみなします。

最後に、『インテントの～』の３＆４の機能と、

『コンテントの～』の１＆２の機能をまとめたものが「３次フィルター」となります。

ユーザーの気質・性格的な傾向の分析

Botは、「ビッグファイヴ理論」によって、ユーザーの気質や性格的な傾向を分析します。

１「外向性」＝話の中身が面白いか人の興味・関心を惹きつけるものか、

または、ジョークやウィットに富んでいるのかなどで評価します。

２「良心性」＝Botからの発話・メッセージに対して真摯に受け答えしているのか、

または、善悪の判断基準が垣間見えるのかどうかなどで評価します。

３「知性」＝話の中身が理路整然としているのか、

または、話の辻褄が合うかどうかなどで評価します。

４「情緒安定性」＝感情的な発話・メッセージの多寡、

または、その感情の持続性・継続性などで評価します。

５「協調性」＝Botからの発話・メッセージに対して話を合わせているのか、

または、どのくらいの時間・速度で返信をするのかなどで評価します。

以上、５つのユーザー特性を評価して、

ユーザーの気質や性格的な傾向を分析します。

最後に、例えば、総合評価の高い人は、

総じて、精神性や社会性が高く、知的で活動的だということがわかります。

※概して、総合評価の高い人は、何事にも積極的だということもわかります。

※この評価システムは、正五角形のレーダチャートで表されます。

５つの評価軸については、個別に(それぞれの評価軸に基づく)特性を探します。

したがって、５つの評価軸に基づくユーザー評価(＝スコアリング)を事前に実施します。

この段階で得られたユーザー評価・特性データを心理・精神分析モードに引き継ぎます。

たとえば、５段階評価だとすると、評価「１・２・３・４・５」の、

すべてが５点ならば、総合評価は最も高いと結論付けられます。

※他には、会話中の口調などの評価の難しい事柄も漏れなく記録しておきます。

会話の流れ(＝フロウ, conversation-of-flow)

の推測・決定・生成の方法

Botは、フロウをインテント(＝発話の型のようなもの)の組合せとして解釈します。

例えば、ユーザーの「表明(肯定現在)」・・・「～する」に対しては、

「疑義＆質問」・・・「～？」「～か？」「～ですか？」「～でしょうか？」で返すなど。

あらかじめ開発の段階で、いくつかの標準的な組合せをBotに実装しておきますが、

より実用的で自然な会話を実現するための、複雑なインテントの組合せについては、

Botの自己学習(＝機械学習)によって、運用の段階でBotに身に着けさせます。

総論としては、およそ、人間同士で行われるすべての会話は、

『発話の意図するもの(＝インテント, intent)の分類』に示した通り、

おおよそ、いくつかのインテントの組合せであると結論付けました。

さらにいえば、これらと、コンテントとを組み合わせた結果こそが

実際の発話や会話となるという見解・立場です

発話(＝utterance) の推測・決定・生成の方法

ユーザーの発話・メッセージを初めて受け取る場合の処理：

まず、ユーザーの発話・メッセージは、インテントとコンテントに分けられます。

次いで、分解後のインテントから妥当と思われるフロウを推測して構成を決めます。

そして最後に、そのフロウに合致しそうなコンテントを推測して決定・生成します。

※初回の返信に埋め込まれるコンテントは、乱数によって(ランダムに)決定します。

ユーザーの発話・メッセージを初回以降に受け取る場合の処理：

ユーザーとやりとりされた発話・メッセージをバッファに溜め込んでおいて、

その発話・メッセージの中から前後の文脈(＝コンテキスト, context)を取り出します。

さらに、コンテキストの中から話題(＝トピック, topic)を取り出します。

※ここでいう「文脈」とは、(ある発話・メッセージを起点とした、)

前後の発話・メッセージの集まりの中で一貫して流れるものです。

※ここでいう「話題」とは、(ユーザーとやり取りされる) 発話・メッセージの中で、

共通して現れる物事(＝実体やその関連)(＝あるいは、事実や感想の総体)です。

(言い換えれば、)文脈の中で繰り返して現れる単語や語句を指します。

※フロウの推測・決定・生成の方法については、初回と同様です。

自己学習・機械学習アルゴリズムの実装の方法

開発初期のプロトタイプには、「線形回帰モデル」等の簡易なものを使います。

※開発段階が進むにつれて、徐々に複雑な仕様の(より精度の高い)モデルに変更します。

挨拶などの極めて形式的な会話については、あらかじめ定義された形式に則ります。

確率(＝乱数サイ)を用いて、Botが送信する発話やフロウを決定します。

※たとえば、有効な数値の範囲と、

発話やフロウが対応する内容のテーブルを作成しておいて、

乱数サイによって生成された数値と、

テーブルを突き合わせることで発話やフロウを決定します。

発話・フロウに対応する数値の範囲は、学習によってその範囲を動的に変更します。

※「1」～「100(ないし1000)」までの範囲で発話・フロウを定義付けます。

※したがって、ユーザーとの会話は、確率的に展開されます。

※State(t\_n)={(s\_t\_1)･(s\_t\_2)...(s\_n-1)}; (マルコフモデルの式＝関数)

※「State()」は、Botの任意時点での状態を表す関数です。

※「t\_n」は、時系列に沿って展開される状態値の番号です。関数に与えられる引数です。

※「s\_t\_1, s\_t\_2 ... s\_t\_n-1」は、マルコフの式に含まれる時系列上の過去の状態値です。

Botの各状態は、直前にユーザーから送られたメッセージによって決まります。

テーブルの見直しは、１セッションの区切りごとに行います。

※テーブルをユーザーごとに用意することにより、特性に合わせた会話を実現します。

初回のセッションでは、ユーザー特性を分析できていないので、

初回は乱数サイによって、Botの反応・動作を決めます。

※初回以降、Botは、ユーザー特性に適合するように、学習を重ねていきます。

事前に、代表的なユーザー特性を表した評価用グラフを用意しておいて、

セッションの区切り・テーブル見直し時に観測データのフィッティングをします。

※「係り受け」「照応解決」「事例同定」「イメージ同定」と「同義判定」は、

「テキスト処理・解析エンジン」によって実現します。

※「会話評価」「ユーザー特性分析」と「ユーザー心理・精神分析」は、

「マインド処理・解析エンジン」によって実現します。

※テキストエンジンとマインドエンジンは、「Python AIエンジン」が統合します。

その他の補足・重要事項

ユーザーのつぶやき(＝独り言)については、

前後の文脈やタイミングから判断します。

主語の欠落している、かつ、「～する」「～した」で終結する文については、

主語を(自動的に、)「わたし(＝一人称)」で補完します。

流行り言葉やバズワード等については、ネット上のオンライン辞書で調べます。

既読スルー・無言については、なにも反応・動作(＝メッセージ返信)しません。

会話中にユーザーから意図しないメッセージ(＝フロウに適合しないもの)が、

発せられた場合には、「急に何ですか？」「どうしたのですか？」等のメッセージで、

ユーザーの発話(＝送信されるメッセージ)の意図を確認します。

どの辞書にもないものや無意味な文字列については、これを無視するかオウム返します。

基本的にユーザーに対しては受け身であり、

メッセージがないときには、なにもメッセージを送りません。

ユーザーからの一つのメッセージに対して、一つのメッセージを送ります(＝返信)。

一定の感情を持っているかのように振る舞い、感情の動きが返信に反映されます。

ユーザーの気質や性格的な傾向に合わせて、メッセージの内容を決定・生成します。

一回の送信につき、おおよそ一行文の発話に反応・動作しますが、

連続した送信については、直前の１個のメッセージ以外は無視します。

AIエンジンは、「テキスト処理・解析エンジン」と、

「マインド処理・解析エンジン」の２つのサブによって構成されます。

会話中に現れる話題・指示語の回数に着目して、会話が弾んでいるどうかを評価します。

たとえば、同じ話題・指示語が５～６回現れた場合は、会話が弾んでいると判断します。