



**POLITECHNIKA
GDAŃSKA**

WYDZIAŁ FIZYKI TECHNICZNEJ
I MATEMATYKI STOSOWANEJ

Imię i nazwisko studenta: Piotr Gronek

Nr albumu: 169534

Studia pierwszego stopnia

Forma studiów: stacjonarne

Kierunek studiów: Fizyka Techniczna

Specjalność: Informatyka stosowana

PROJEKT DYPLOMOWY INŻYNIERSKI

Tytuł projektu w języku polskim: Narzędzie wykorzystujące algorytmy uczenia maszynowego do wstępnej oceny kandydatów do pracy na stanowiskach kierowniczych

Tytuł projektu w języku angielskim: Implementation of the system for pre-screening of the candidates for the management positions by means of machine learning algorithms

Potwierdzenie przyjęcia projektu	
Opiekun projektu	Kierownik Katedry/Zakładu (pozostawić właściwe)
<i>podpis</i>	<i>podpis</i>
dr hab. inż. Marta Łabuda	

Data oddania projektu do dziekanatu:

Streszczenie

Celem pracy było utworzenie aplikacji służącej jako bot przeprowadzający rozmowy o pracę na stanowisko menadżera, przy użyciu algorytmów wykorzystujących możliwości uczenia maszynowego. Pomysł powstał we współpracy z dr. Maciejem Karpiczem, który jest konsultantem projektu inżynierskiego. Jako wzorzec technologiczny została mi zaproponowana upubliczniona aplikacja wykonana przez firmę D-labs^[1] na Hackathonie^[2] w 2019 roku. Problem, z którego wywodzi się projekt, jest regularnie napotykanym przez aplikantów do pracy, wiąże się on z ogromną ilością otrzymywanych przez rekruterów życiorysów, oraz brakiem odpowiedniej ilości czasu potrzebnego na weryfikację aplikacji. Został on potwierdzony poprzez przeprowadzenie wywiadów z anonimowymi rekruterami. Programista przydzielony do zajmowania się rekrutacją wypowiedział się, że średnio sprawdzenie pojedynczego CV zajmuje mu około jednej minuty. Osoba zajmująca się ściśle rekrutacją oznajmiła, że średnio trzy minuty są jej potrzebne na stwierdzenie czy dalsza weryfikacja aplikacji jest potrzebna. Na podstawie zebranych informacji powstała teza, że często tylko część treści składanych dokumentów jest sprawdzana, co często z powodu złego sformatowania naszego CV powoduje ominięcie ważniejszych elementów. Projekt ma na celu redukcję czasu poświęcanego na sprawdzenie dokumentów przez rekruterów. Pomagając zwłaszcza tym, których zadaniem w pracy nie jest zajmowanie się tylko i wyłącznie zasobami ludzkimi. Dzięki chat-botowi^[3] przeprowadzającemu skrupulatnie rozmowę o pracę. Bazuje on na systemie przyznawania za odpowiedzi punktów, zweryfikowanym z rekruterami. Tylko aplikanci z odpowiednim progiem punktowym trafią do dalszej weryfikacji przez firmę. Aplikacja została utworzona w języku Python^[4], operując na bibliotekach konwertujących mowę na tekst zrozumiałą dla aplikacji, oraz tekst na mowę słyszalną przez osobę, z którą przeprowadzana jest rozmowa. Za pośrednictwem systemów Google API^[5], powstała część uniwersalnych pytań, zbierających informacje na temat aplikanta. Pozostałe pytania operują na logice wychwytywania słów bądź zdań kluczy, odpowiednich do tych zadanych wcześniej aplikacji. Sprawdzając wartości poprawności wypowiedzianych słów, wybierana jest prawdopodobna odpowiedź, o ile taka istnieje. Po ukończeniu budowy aplikacji zrozumiano, jak duże ma ona możliwości rozwoju. Udoskonalenie systemu pytań, żeby na ich podstawie aplikacja reagowała odpowiednio do danych odpowiedzi, rozszerzenie jej na inne dziedziny pracy, bądź ulepszenie sposobu wypowiadania się programu, żeby dawał wrażenie bardziej naturalnego. Możliwości rozwoju są nieograniczone, a samo wdrożenie tego typu rozbudowanej aplikacji do miejsca pracy znacząco poprawiłoby proces weryfikacji aplikantów.

Słowa kluczowe: APLIKACJA, APLIKANT, BOT, INFORMATYKA, REKRUTACJA, ROZMOWA.

Dziedzina nauki i techniki, zgodnie z wymogami OECD: inżynieria informatyczna, Sprzęt komputerowy i architektura komputerów

Abstract

The Goal of the work was to create an application conducting interviews to select candidates for the managerial position, with algorithms using machine learning capabilities. The idea was created in cooperation with dr. Maciej Karpicz, who is an engineering project consultant. As a technological template, I was offered a public application made by D-labs at Hackathon in 2019. The problem from which the project originates is regularly encountered by applicants for work, it is associated with the huge amount of CVs received by recruiters, and the lack of the right amount of time needed to verify the application. It was confirmed by conducting interviews with anonymous recruiters. The programmer assigned to deal with recruitment said that, on average, checking a single CV takes about one minute. A person dealing strictly with recruitment announced that they need an average of three minutes to determine whether further verification of the application is needed. Based on the collected information, the thesis arose that often only part of the content of submitted documents is checked, which often due to the wrong formatting of our CV causes bypassing more important elements. The project aims to reduce the time spent on recruiters checking documents. Helping especially those whose job at work is not only about human resources. Thanks to the chat-bot conducting a meticulous job interview. It is based on a system of awarding points for answers, verified with recruiters. Only applicants with the appropriate point threshold will go to further verification by the company. The application was created in Python, using libraries that convert speech into text understandable for the application, and text to speech heard by the person being interviewed. Through the Google API systems, some universal questions that gathers information about the applicant were created. Other questions operate on the logic of capturing words or key phrases relevant to those previously asked by application. When checking the correctness of the words spoken, the probable answer is chosen, if any. After completing the construction of the application, it was understood how much development opportunities it has. Improving the system of questions, so that on their basis the application responds appropriately to the given answers, expanding it to other areas of work, or improving the way the program speaks so that it gives a more natural impression. The development possibilities are unlimited, and the very implementation of this type of extensive application to the workplace would significantly improve the verification process of applicants.

Keywords: APPLICATION, APPLICANT, BOT, IT, RECRUITMENT, CONVERSATION.

Spis treści:

STRESZCZENIE.....	3
ABSTRACT.....	4
WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW	6
WSTĘP.....	7
1. CEL PROJEKTU.....	8
1.1 Problematyka projektu.....	8
1.2 Idea rozwiązania problemu.....	8
1.3 Zadania przeprowadzone w celu wykonania pracy.....	9
2. SPECYFIKACJA PROJEKTU.....	10
2.1 Opis ogólny.....	10
2.1.1 Komponenty programu.....	10
2.1.2 Interfejsy systemowe.....	11
2.1.3 Interfejs użytkownika.....	11
2.1.4 Interfejs oprogramowania.....	11
2.1.5 Interfejs sprzętowy.....	12
2.1.6 Interfejs komunikacyjny.....	12
2.2 Funkcja aplikacji.....	12
3. OPIS APLIKACJI PROJEKTU.....	13
3.1 Logika i opis działania aplikacji.....	13
3.2 Funkcje zawarte w aplikacji.....	15
3.2.1 Funkcja parsująca tekst.....	15
3.2.2 Funkcja parsująca tekst oparta o jednostki wielowyrzowe.....	16
3.2.3 Funkcje operujące na zwróconym kluczu.....	16
3.2.4 Funkcje operujące na wyrażeniach regularnych.....	17
3.2.5 Funkcje konwertujące treść wypowiedzi TTS i STT.....	18
3.3 Najważniejsze biblioteki i technologie.....	19
3.3.1 Biblioteka DeepPavlov	19
3.3.2 Moduł os.....	20
3.3.3 Pakiet stringdist.....	20
3.3.4 Moduł re.....	20
3.3.5 Klasa opakowująca Google Maps.....	20
3.3.6 Moduły pyaudio, wave, io, os.....	20
3.3.7 Implementacja Google Cloud.....	20
4. TESTOWANIE APLIKACJI.....	21
5. PODSUMOWANIE.....	22
Wykaz literatury.....	23
Dodatek A. Spis zawartości dołączonej płyty CD	26

Wykaz ważniejszych oznaczeń i skrótów.

- D-Labs - Nazwa firmy, do której należy doradca pracy inżynierskiej dr Maciej Karpicz
- CV (Curriculum vitae) - życiorys zawodowy. Dokument składany przez aplikanta na dane stanowisko.
- Google API key - klucz zapisywany w pliku ".json" potrzebny do połączenia z chmurą google
- BERT (Bidirectional Encoder Representations from Transformers) - Głównym zadaniem BERT jest lepsze zrozumienie intencji wyszukujących, a przede wszystkim prawidłowe dopasowanie wpisywanych słów kluczowych.
- Regex - wyrażenie regularne, symboliczny zapis znaków, które mogą wystąpić w oczekiwanym słowie bądź zdaniu
- Hardware - materialna część komputera
- Chat-bot - program symulujący rozmowę z człowiekiem
- Microsoft Windows 10 - System operacyjny, na którym utworzona została aplikacja
- IDE pyCharm (Integrated Development Environment) - zintegrowane środowisko programistyczne, w którym została utworzona aplikacja, utworzone przez firmę JetBrains
- TTS (TextToSpeech) - metoda odpowiedzialna za zamianę tekstu na dźwięk słyszany przez użytkownika w postaci głosu
- STT (SpeechToText) - metoda odpowiedzialna za zamianę nagrania odpowiedzi użytkownika na tekst wykorzystywany w aplikacji
- .wav (wave form audio) - typ rozszerzenia pliku, odpowiedni dla aplikacji format plików dźwiękowych
- .py - typ rozszerzenia pliku odpowiedni dla Pythona
- Odległość Levenshteina - miara odmienności znaków w porównywanym tekście, wyliczana poprzez ilość różnic pomiędzy słowem porównywanym oraz kluczem
- Słownik - struktura danych w Pythonie, operująca na zbiorze elementów bazująca na kluczach do nich przypisanych
- Parsowanie - analizowanie ciągu znaków w celu ustalenia jego struktury
- Jednostka wielowyrazowa (multi word entity) - Wyrażenie używanego do określenia typu wyszukiwanego elementu tekstu
- NER (named entity recognition) - metoda polegająca na identyfikacji i wyłanianiu pożądanych wyrażeń
- SSML (Speech Synthesis Markup Language) - język znaczników służący do opisu interaktywnych dialogów pomiędzy człowiekiem i komputerem

Wstęp

Rozwój technologii komputerowej, oraz algorytmów odpowiadających za sztuczną inteligencję, pozwala ludziom na rozszerzanie swoich możliwości zarządzania czasem. Mimo tych faktów nadal muszą optymalizować wiele istotnych zadań na ich niekorzyść, bądź na niekorzyść osób, dla których je wykonują. Występowanie takowej sytuacji pozostawia szczególnie negatywny wydźwięk w dziedzinie zarządzania zasobami ludzkimi. Każdy zasługuje, aby wizytówka, którą jest dokument CV, została sumiennie i bez pośpiechu sprawdzona.

Dlatego też, w niniejszej pracy podjęto się utworzenia aplikacji symulującej przebieg rozmowy o pracę, mającej na celu przeprowadzić rekrutację, zgodnie z wyznaczonymi przez rekrutera wytycznymi, wraz z ocenieniem umiejętności aplikanta, w określonej wcześniej skali. Program prototypowy jest obsługiwany aktualnie na platformie Windows^[6] posiadającej odpowiednie sterowniki. Wymaga, w celu działania, spełnienia odpowiednich warunków wymienionych w niniejszej pracy. Finalnie po odpowiednich modyfikacjach system, może zostać umieszczony w wirtualnym centrum telekomunikacyjnym, bądź na stronie internetowej należącej do firmy zainteresowanej wprowadzeniem e-selekcji aplikantów.

Technologia chat-botów, sztucznej inteligencji lub aplikacji predefiniowanych imitujących rozmowy międzyludzkie w XXI wieku jest jedną z najbardziej prężnie rozwijających się dziedzin. Nie brakuje ich w miejscach pracy jako elementy przekierowujące rozmówcę, który próbuje skontaktować się z danym działem. W dziedzinie rekrutacji na stanowiska pracy jednak nie występują, żadne większe osiągnięcia w tej dziedzinie. Wnioskuje, że jest to wywołane niechęcią ludzi do przeprowadzania tak istotnych rozmów z maszyną zamiast człowieka. Jednak przy bieżącym rozwoju technologii jest to kwestia czasu, zanim udogodnienia typu niniejszego projektu, odnajdą swoje miejsce na rynku pracy jako obowiązkowa aplikacja ułatwiająca zawód zarządzania zasobami ludzkimi.

W pierwszym rozdziale pracy zawarto cel projektu, jego problematykę oraz wykonane zadania. Wraz z częścią informacji zebranych w przeprowadzonym wywiadzie. Drugi rozdział poświęcony jest specyfikacji projektu aplikacji. Zostały w nim określone rodzaje interfejsów, wraz z opisem ogólnym oraz przewidzianą funkcjonalnością oprogramowania. Trzeci rozdział poświęcony jest przebiegowi działania aplikacji. Wymienione w nim zostały najważniejsze metody wraz z instrukcją ich funkcjonalności, oraz ze skróconym opisem najważniejszych bibliotek, które zostały w niej wykorzystane. W czwartym rozdziale zawarty został opis przebiegu testów przeprowadzanych w trakcie pracy nad projektem.

1. Cel projektu

Celem niniejszej pracy było utworzenie aplikacji, mającej za zadanie przeprowadzać z użytkownikiem rozmowę sprawdzającą jego kwalifikacje na stanowisko kierownika projektu. Ułatwiając klasyfikację aplikantów na tych, którzy osiągnęli odpowiedni wynik. Nie ma ona na celu zastąpienia osoby weryfikującej CV, lecz jedynie pomoc w zmniejszeniu ilości dokumentów. Pozostawiając tylko te, które są zadowalające i wymagają dalszego etapu sprawdzenia ręcznego. Profil zawodu menadżera/kierownika został zaproponowany przez doradcę pracy inżynierskiej. Mogłaby ona przeprowadzać selekcje do dowolnego zawodu po odpowiedniej modyfikacji. Jednak szybkie sprecyzowanie dziedziny było wymagane w celu ograniczenia problematyki projektu i uproszczenia zbierania informacji na temat prowadzenia rozmów selekcyjnych.

1.1 Problematyka projektu

Problem, którego dotyczy poniższa praca, odnosi się do braku możliwości poświęcania odpowiedniej ilości czasu na sprawdzanie dokumentów aplikantów. Został on potwierdzony w wywiadzie z anonimowymi rekruterami. Informacje te zostały zebrane przed rozpoczęciem tworzenia aplikacji, treść ich streszczona zostanie w tym podpunkcie. Zgodnie z tymi informacjami, czas poświęcany na przeczytanie dokumentu CV przez osobę weryfikującą to średnio jedna minuta do trzech minut. W zależności od preferencji osoby sprawdzającej aplikacje weryfikowane są najczęściej na dwa sposoby.

- Pierwszym z nich jest sposób techniczny, w którym rekruter nie poświęca większej uwagi na estetykę i mniej znaczące jej punkty. Najbardziej interesuje go pierwsza strona, na której oczekuje wykazu najważniejszych informacji. Należą do nich doświadczenie w zawodzie, największe osiągnięcia dotyczące pracy, ewentualne wykształcenie wyższe związane z zawodem. W wypadku niespełnienia tych wcześniej określonych wymogów aplikant, może zostać natychmiastowo skreślony.
- Drugi sposób jest bardziej ogólny, dotyczy rekruterów zwracających uwagę na osobowość osoby aplikującej. Doszukują się oni informacji często niezawartych w CV, lecz bardzo dla nich istotnych. Najlepiej wyraża to kwestia zainteresowania powodem zmiany miejsca pracy, rekruter zadaje sobie pytanie, czy aplikant odszedł z powodu wywołanego przez firmę? Analizuje czy pracownik nie był w stanie pracować przez warunki panujące w środowisku pracy. Na tej podstawie jest w stanie stwierdzić czy dany aplikant będzie w stanie odnaleźć się w nowym miejscu pracy.

Obydwie te metody, jak i wszelkie inne preferencje osób weryfikujących, odnośnie sprawdzania są prawidłowe. Ograniczone jednak ilością składanych dokumentów. Jeżeli aplikant nie utworzy swojej aplikacji w sposób odpowiadający osobie sprawdzającej, jego szanse mogą drastycznie zmaleć przez ominięcie ważniejszych kwestii w niej zawartych.

1.2 Idea rozwiązania problemu

Przy odpowiedniej ilości informacji dotyczących firmy rekrutującej, jak i preferencji osoby sprawdzającej da się utworzyć chat-bota przeprowadzającego rozmowę z aplikantem. Wzorcem przebiegu aplikacji może być uproszczona rozmowa o pracę z aplikującym który przeszedł już etap sprawdzania dokumentu CV w danej organizacji. Procesem utworzenia takiego programu jest zebranie zbioru pytań wraz z określeniem ich wag, zgodnie z którymi przyznawane są punkty. W ten sposób np. z 300

otrzymanych przez rekrutera CV, mającego wysokie oczekiwania odnośnie pracownika, trafić może 40 z nich. Wraz z taką ilością aplikacji pracownik zarządzający zasobami ludzkimi nie tylko oszczędza czas, ale ma też możliwość dokładniejszej selekcji i wybrania najlepszych kandydatów przed zaproszeniem ich na rozmowę kwalifikacyjną. W celu prawidłowego działania takiego procesu nie należy rezygnować z dokumentów CV, jako że są one elementem weryfikującym prawdomówność aplikanta w rozmowie z programem. Wszelakie nieszczerości aplikującego, bądź błędy wynikające z rozmowy, powodujące zawyżony wynik otrzymany przez aplikanta zostałyby przez osobę weryfikującą owy dokument sprawdzone i zostałyby ona odrzucona. W zależności od preferencji danego przedsiębiorstwa program ten mógłby zostać zaimplementowany w procesie rekrutacji online na stronie internetowej bądź uruchamiany na zbudowanym wirtualnym centrum telekomunikacyjnym w celu prowadzenia rozmów telefonicznych. Z powodów technicznych musi również występować możliwość braku akceptacji przeprowadzonej rozmowy przez samego rozmówcę, oraz możliwość jej powtórzenia. W wypadku gdy rozmowa zostanie ukończona, aplikant otrzymuje spis pytań oraz odpowiedzi, które złożył. Na ich podstawie, może się okazać, że niewyraźna mowa, bądź jakość dźwięku otrzymanego przez program spowodowała błąd w rozmowie. Niewychwycony błąd tego typu nie wywołał powtórzenia pytania, co powoduje niepoprawną finalną ocenę aplikanta.

1.3 Zadania przeprowadzone w celu wykonania pracy

W trakcie prac nad projektem wymagane było ustalenie priorytetów, w celu prawidłowego przebiegu pracy. Zaczęto od skontaktowania się z wybranymi pracownikami zajmującymi się rekrutacją. Wyrazili oni zgodę na anonimową współpracę w projekcie. Udzielili informacji na temat przebiegu procesu rekrutacji w środowisku ich pracy, wraz z opisem sposobów sprawdzania dokumentów składanych przez aplikujących. Podali także i uzasadnili sposób oceniania poszczególnych elementów CV. Na podstawie zebranych informacji powstała logika przebiegu aplikacji oraz system wag odpowiadających poszczególnym pytaniom. Posiadając podstawę do zbudowania systemu, zapoznano się z aplikacją wzorcową^[7], oraz wykorzystywanymi przez nią bibliotekami. Wraz z rozwojem pracy odbywały się testy prowadzące do utworzenia niniejszego projektu pracy inżynierskiej.

2. Specyfikacja projektu

W tym rozdziale pracy opisana zostanie specyfikacja aplikacji, nad którą prowadzona była praca. Jako że aplikacja została ukończona na etapie, w którym dostęp do niej możliwy jest jedynie poprzez środowisko programistyczne, zwrot “użytkownik” wykorzystywany w tym rozdziale symbolizować będzie programistę, bądź osobę mającą możliwość uruchomić aplikację z poziomu kodu źródłowego.

2.1 Opis ogólny

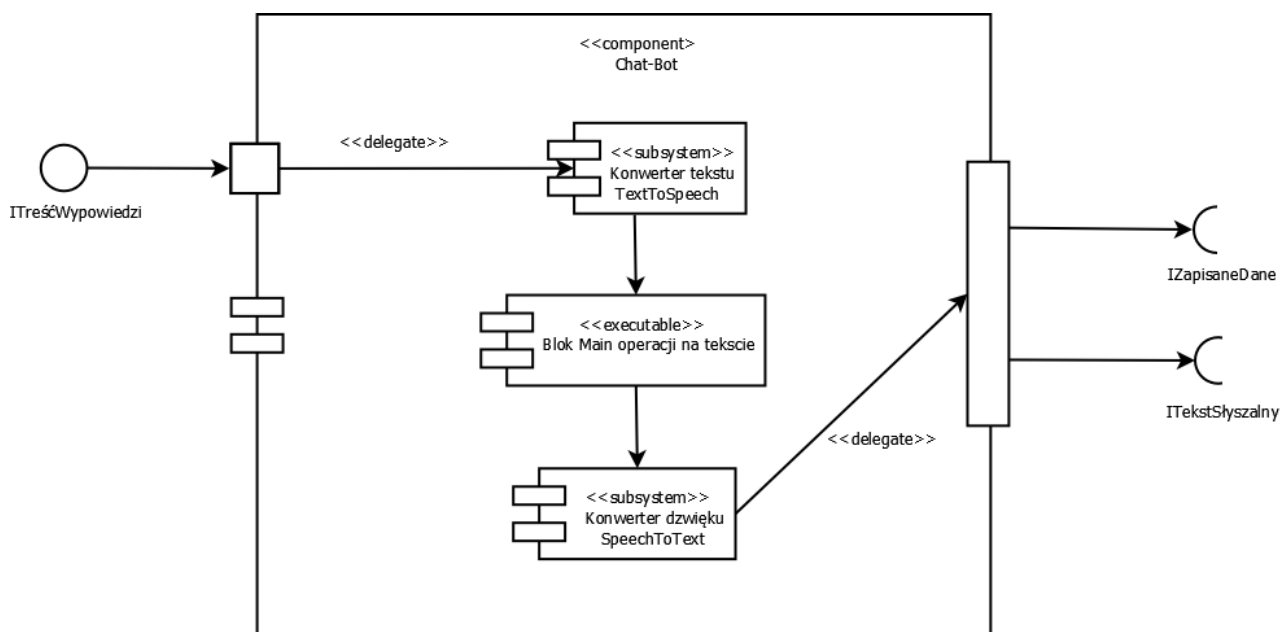
Utworzone oprogramowanie bazuje na prostej logice pytań i odpowiedzi, z możliwościami rozszerzenia jej przebiegu na wiele dziedzin zawodowych. Po uruchomieniu aplikacji użytkownik otrzymuje informację dźwiękową informującą o celu przeprowadzanej rozmowy. Dzięki operowaniu na odpowiednich bibliotekach system przeprowadza konwersje predefiniowanych zdań na dźwięk słyszalny przez użytkownika imitujący ludzki głos. Następnie celem użytkownika jest zrozumiałe, odpowiednio głośne, odpowiadanie na słyszane pytania w wyznaczonym limicie czasu. Czas ten jest zoptymalizowany do umożliwienia odpowiedzi jednostkowej, bądź pełnym zdaniem, mierzony jest od momentu ukończenia zadawania pytania. Finalizując rozmowę, użytkownik otrzymuje informacje na temat zdobytej ilości punktów oraz wykaz odpowiedzi na poszczególne pytania.

2.1.1 Komponenty programu

Utworzona aplikacja wzorowała się na chat-bocie umawiającym na wizyty do gabinetu lekarskiego. Mimo ilości aplikacji będących alternatywą do współpracy człowieka z komputerem jest ona unikatowa z powodu dziedziny, której dotyczy. Jak pokazano na poniższym diagramie 2.1.1 w skład komponentu “Chat-Bot” aplikacji wchodzi:

- Podsystem konwertujący otrzymaną treść wypowiedzi
- Część wykonywalna, czyli “Main”, w której przeprowadzane są operacje
- Podsystem konwertujący tekst, przekazujący treść słyszalną, oraz dane do zapisu

Typ budowy tego komponentu jest uniwersalny dla chat-botów.



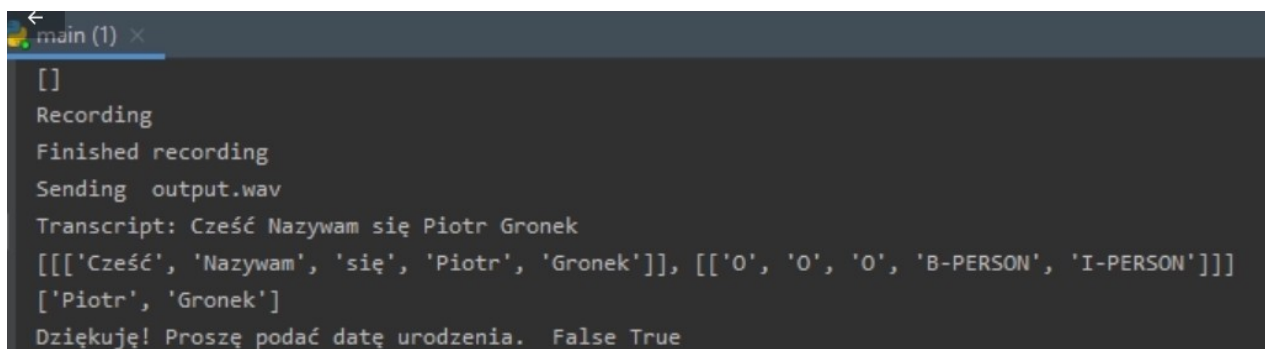
Rysunek 2.1.1 Diagram komponentów aplikacji

2.1.2 Interfejsy systemowe

Środowiskiem funkcjonowania oprogramowania jest Microsoft Windows 10, w wersji prototypowej bez utworzonego zewnętrznego interfejsu użytkownika. Jest on plikiem wykonywalnym utworzonym w pythonie za pośrednictwem zintegrowanego środowiska programistycznego pyCharm^[8]. Wymogiem działania aplikacji jest implementacja odpowiednich bibliotek do środowiska, bądź w przyszłości wyeksportowanie ich do osobnej aplikacji wykonywalnej. Warunkiem pełnego działania bibliotek jest połączenie z interfejsem Google API za pośrednictwem internetu oraz posiadanie odpowiednich sterowników karty graficznej Nvidia^[9]. Finalnym zadaniem byłoby umieszczenie w sieci wyeksportowanej aplikacji, na urządzeniu spełniającym warunki systemowe odpowiadającym za serwer.

2.1.3 Interfejs użytkownika

Interfejs użytkownika, obejmuje konsolę IDE pyCharm, w której wyświetlają się informacje odnośnie przebiegu działania aplikacji. Na poniższym rysunku 2.1.3 pokazano wycinek konsoli z działającego programu symbolizującą aktualny interfejs użytkownika.



```
<
main (1) x
[]
Recording
Finished recording
Sending output.wav
Transcript: Cześć Nazywam się Piotr Gronek
[[['Cześć', 'Nazywam', 'się', 'Piotr', 'Gronek']], [['O', 'O', 'O', 'B-PERSON', 'I-PERSON']]]
['Piotr', 'Gronek']
Dziękuję! Proszę podać datę urodzenia. False True
```

Rysunek 2.1.3 Obraz wyświetlenia konsoli

Zgodnie z przebiegiem działania programu użytkownikowi wyświetlają się kolejno informacje. Na temat rozpoczęcia nagrywania głosowego po zadanych pytaniu, oraz o ukończeniu nagrywania. Następnie treść wypowiedzi jest przesyłana do pliku “output.wav”, na którym przeprowadzane są operacje konwersji. Po zakończeniu konwersji użytkownikowi ukazują się tekst, przedstawiający jego wypowiedź. W następnym kroku następuje rozbiecie wypowiedzi na poszczególne elementy oraz umieszczenie ich w zbiorze. Przypisany do każdego z nich zostaje klucz, w tym wypadku jest to klucz *imienia i nazwiska*. Zatem wszelkie elementy niebędące nimi oznaczane są jako *inne elementy* oznaczone [O]. Wykryte *imie i nazwisko* użytkownika zostaje zapisane zgodnie z określonym wcześniej standardem [B-Person] oraz [I-Person] symbolizującymi pierwszy wykryty wyraz, oraz następny, odpowiadający kluczowi. Przed ukończeniem aktualnego obiegu wyświetla się wynik dopasowanego klucza, a następnie odtwarzana oraz wyświetlana jest wiadomość symbolizująca koniec obiegu programu. W trakcie działania programu w tym oknie wyświetlane są również treści związane z wszelkiego rodzaju nieokreślonymi wcześniej błędami, których wystąpienie powoduje koniec działania aplikacji.

2.1.4 Interfejs oprogramowania

Oprogramowanie złożone jest z plików współdziałających ze sobą. Główny plik, w którym znajduje się kod aplikacji, złączony jest odpowiednio z plikiem “TextToSpeech” odpowiedzialnym za konwersję tekstu na dźwięk symulujący mowę. “SpeechToText”, w którym znajdują się metody konwertujące dźwięk z

połączonego z plikiem nagrania "output.wav", wcześniej zapisanego przez aplikację. Oraz "data.dat" generowany po zakończonej rozmowie odpowiadający za kontener na dane, które zostały przypisane do zmiennych w trakcie zadawania pytań użytkownikowi.

2.1.5 Interfejs sprzętowy

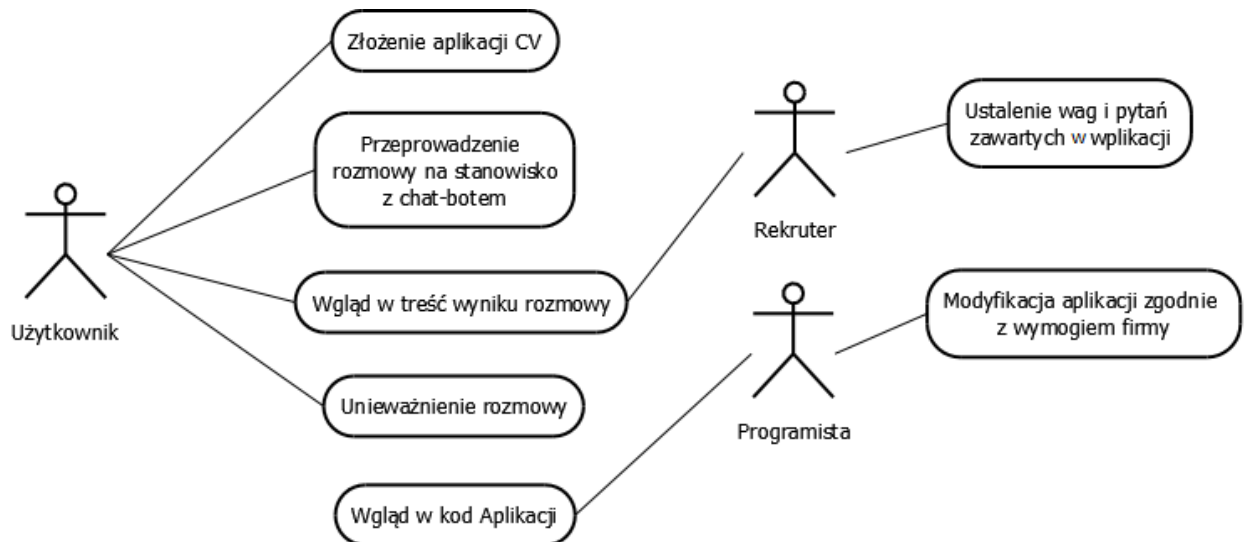
Interfejs oprogramowania działa obsługując urządzenia dźwiękowe za pomocą odpowiednich bibliotek. Dzięki nim aplikacja odbiera treść wypowiedzianą przez użytkownika do mikrofonu, oraz wysyła odpowiednią treść dźwiękową do głośnika. Dane przetwarzane przez aplikację są typu tekstowego bądź typu ".wav" odpowiednio konwertowane w zależności od aktualnie wymaganego przez aplikację formatu.

2.1.6 Interfejs komunikacyjny

Aplikacja komunikuje się poprzez łącze internetowe z chmurą Google, na której znajduje się wykaz działania algorytmów konwertujących, oraz wyszukujących adres. Odbywa się to dzięki zatwierdzonemu przez system oraz Google odpowiedniemu kluczowi aplikacji Google API. Bez połączenia z siecią aplikacja nie jest w stanie prawidłowo działać.

2.2 Funkcja aplikacji

W tym podpunkcie opisana zostanie funkcjonalność aplikacji prototypowej oraz jej zastosowanie, także idea działania finalnego systemu. Funkcjonalności programistyczne aplikacji zostaną wymienione w rozdziale dotyczącym przebiegu działania aplikacji.



Rysunek 2.2 Diagram przypadków użycia finalnej aplikacji

Zgodnie z powyższym diagramem 2.2A. Działanie aplikacji ma maksymalizować zaangażowanie użytkownika, czyli aplikanta, w przebieg pracy. Osoba rekrutująca po ustaleniu wymagań zgodnych z jej oczekiwaniami wymagana byłaby jedynie do sprawdzania dokumentów, które przeszły przez program. Natomiast programista wystarczy, że zmodyfikuje treść pytań, kluczy, oraz przyznawane za nie punkty zgodnie z oczekiwaniami firmy. Miałoby to na celu optymalizację zarządzania czasem procesu rekrutacji.

Wedle działania utworzonego prototypu, użytkownik, który spełnia wszelkie wymagania sprzętowe, jest w stanie uruchomić aplikację. Następnie po zaktualizowaniu odpowiednich bibliotek oraz połączeniu z serwerem Google cloud słyszy, oraz widzi w konsoli programu wiadomość powitalną. Podążając zgodnie z wytycznymi wymaganymi do prawidłowego przebiegu rozmowy, powinien on być w stanie przeprowadzić rozmowę, oraz ukończyć ją. W przypadku wystąpienia różnic literowych w słowach podobnych aplikacja

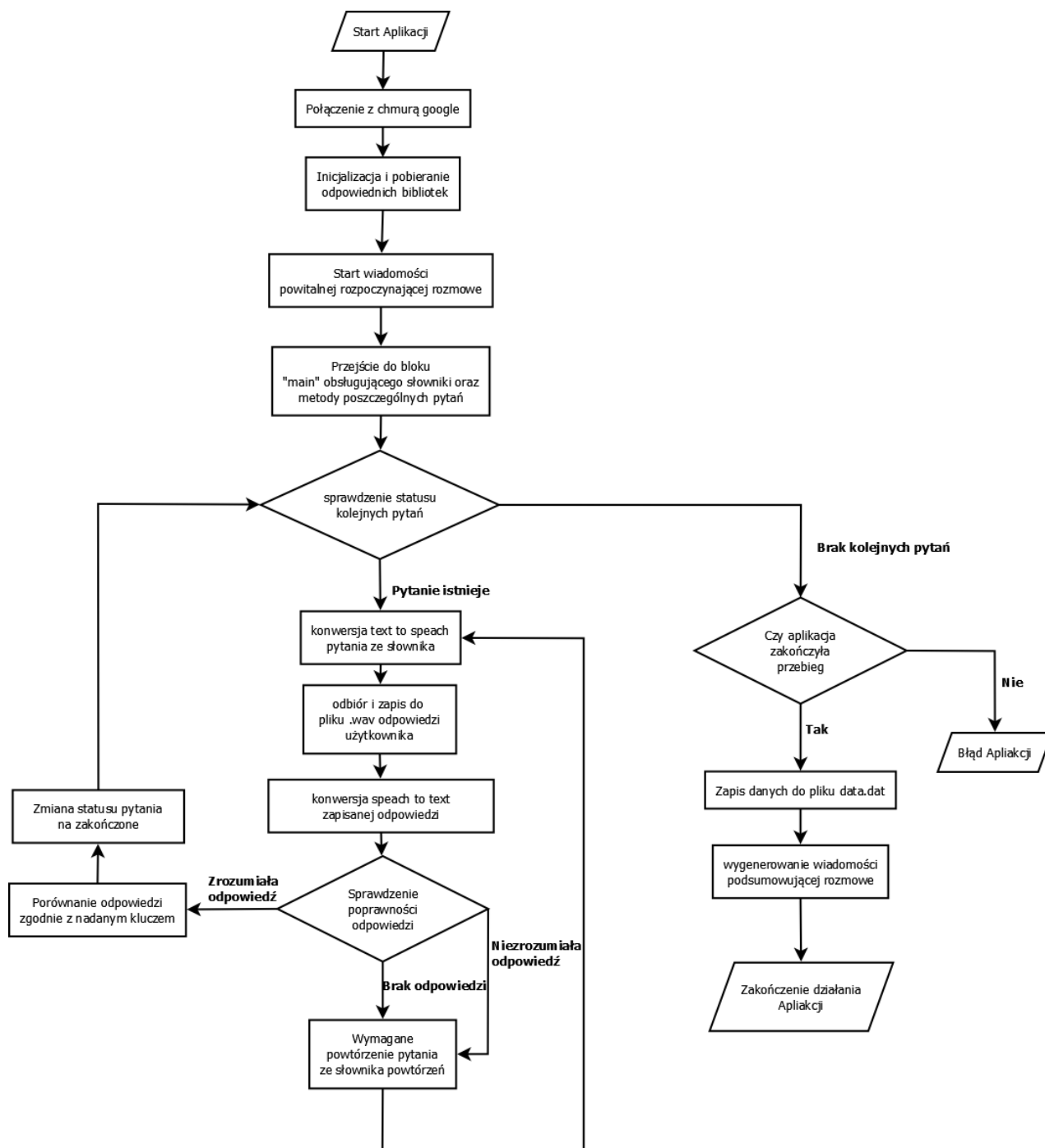
powinna znaleźć najlepsze dopasowanie, natomiast gdy wiadomość będzie zbyt cicha lub niesłyszalna zwrócić użytkownikowi opisującą to informacje i powtórzyć zadane pytanie. Na koniec powinien otrzymać i wynik procentowy ilości punktów, które zdobył oraz mieć wgląd do pliku z zapisem danych jego rozmowy.

3. Opis aplikacji projektu

W tym rozdziale zostanie opisane działanie aplikacji, jej funkcję oraz wykorzystane technologie. Po etapie zapoznawczym z projektem aplikacji Hacathonowej nastąpiła implementacja projektu. Dużym wyzwaniem w tworzeniu projektu była nauka nieznanymi wcześniej bibliotek odpowiedzialnych za przeprowadzanie operacji na tekście i treści dźwiękowej, wspomagających konstrukcje chat-botów. Także samo rozwinięcie umiejętności programowania w języku Python.

3.1 Logika i opis działania aplikacji

Na poniższym diagramie 3.1 zademonstrowano szkic pojedynczego uruchomienia aplikacji.



Rysunek 3.1 Diagram przebiegu aplikacji

Po uruchomieniu program łączy się z chmurą Google za pomocą Google API key, w celu późniejszego korzystania z bibliotek Google odpowiadających za konwersje tekstu do dźwięku oraz dźwięku do tekstu. Pozwala ona także na używanie używania map Google, potrzebnych do sprawdzenia poprawności występowania adresu podanego przez rozmówcę. Następnym krokiem aplikacji jest inicjalizacja odpowiednich bibliotek oraz pobranie wymaganej wersji BERT^[10], czyli algorytmu odpowiadającego za poprawne wyszukiwanie i dopasowywanie słów do wcześniej zdefiniowanych słowników kluczy. Kolejnym etapem działania aplikacji jest jej stan gotowości, użytkownik otrzymuje treść wiadomości powitalnej aplikacji za pomocą konwersji tekstu na mowę słyszalną, przekazując otrzymaną wiadomość użytkownikowi. Następnie następuje początek operacji na słownikach, na których oparta jest cała logika aplikacji. Każdy blok rozmowy posiada odpowiednio:

- Status - zmieniający się w zależności od tego czy pytanie zostało prawidłowo zakończone, czy też nie.
- Funkcje - które obsługują dany blok. Wywołują metody porównywania, operując na otrzymanej odpowiedzi, oraz działają na wcześniej utworzonym systemie wag. Przyznając punkty poprzez zmienną odpowiedzialną za ich ilość. Punkty przyznawane są na podstawie dopasowań wypowiedzi do słów kluczy.
- Parametry - czyli predefiniowane zbiory słów kluczy wymagane do porównań z wypowiedzią.
- Etap pytania - zawierający treść pytania, które ma usłyszeć użytkownik.
- Podsumowanie tekstu - czyli zapis całkowity wypowiedzi użytkownika, na którym przeprowadzane są późniejsze operacje porównawcze.
- Etap powtórzenia - zawierający treść pytania, które ma zostać powtórzone, z ewentualnym wyjaśnieniem. Wywoływany w przypadku gdy odpowiedź została przez aplikację nie zrozumiana, bądź nie było wystarczająco głośna.

W pierwszej kolejności aplikacja sprawdza, czy pytanie już się nie odbyło, jeżeli jego status jest fałszywy, następuje całkowity przebieg pytania. Zadanie pytania użytkownikowi, następnie odsłuchanie w odpowiednim czasie jego wypowiedzi. Zapis wypowiedzi do pliku ".wav,", na którym przeprowadzana jest operacja konwersji dźwięku na dane zrozumiałe przez aplikację. Po konwersji następuje sprawdzenie, czy dana treść została zrozumiana. Jeżeli jest ona pusta bądź nie spełnia wymagań dotyczących porównania dystansu słów przez algorytm, zostaje zwrócona informacja o nieodpowiedniej wypowiedzi i wywołany etap powtórzeń. W przeciwnym wypadku odpowiednio elementy wypowiedzi bądź całe zdania zostają według odpowiadających im funkcji, dopasowywane do klucza przyznając punkty za prawidłowe zestawienia. Treść wypowiedzi zostaje zapisana dla danego bloku w celu późniejszego wykorzystania. Po zakończeniu danego bloku następują po nim kolejne przebiegające w ten sam sposób, dopóki aplikacja nie osiągnie braku pytań. Następnie aplikacja sprawdza, czy przejście do finalnej części nastąpiło po ukończeniu ostatniego bloku, czy też zostało to wywołane błędem. Przechodząc do ostatniego kroku po pozytywnej weryfikacji pokonanych etapów, program zbiera podsumowane dane. Przypisuje je odpowiednim elementom i zapisuje je do pliku danych aplikanta. Następnie generuje finalną odpowiedź, na którą składa się wynik, który otrzymał, oraz podziękowanie za przeprowadzoną rozmowę.

3.2 Funkcje zawarte w aplikacji

W tym podpunkcie wymienione zostaną najważniejsze funkcje, na których opiera się działanie aplikacji. Wyjaśniony zostanie przebieg ich działania, bez dokładnego zagłębiania się w treść kodu. W celu zrozumienia wartości występujących w funkcjach wyjaśnione zostaną następujące oznaczenia

- `Input_text` - niemodyfikowana, przekonwertowana na tekst, treść wypowiedzi użytkownika, na której operują funkcje.
- `List_of_words` - słownik słów kluczy, do których przyrównywane są odpowiednie elementy wypowiedzi użytkownika.
- `Desirable_dist` - wartość określająca maksymalną dozwoloną miarę odległości znaków, między oczekiwaną formą klucza a danym elementem wypowiedzi.

3.2.1 Funkcja parsująca tekst

Poniższy fragment kodu na rysunku 3.2.1 jest jedną z trzech funkcji operujących na tekście otrzymanym po wypowiedzi użytkownika.

```
def parse_text_with_points(input_text, list_of_words, desirable_dist):
    global value_of_dist, min_dist
    correct_answers = 0
    input_text = input_text.strip().lower()
    print(input_text)
    list_input = list(input_text.split(" "))
    print(list_input)
    for i in list_input:
        value_of_dist = [stringdist.levenshtein(i, word) for word in list_of_words]
        print(value_of_dist)
        min_dist = min(value_of_dist)
        if min_dist < desirable_dist:
            correct_answers += 1
        print(correct_answers)
    print(list_of_words[value_of_dist.index(min_dist)])
    return True, correct_answers
```

Rysunek 3.2.1 Treść funkcji parsującej tekst

Treść, którą przetwarza aplikacja, na początku zostaje znormalizowana poprzez usunięcie ewentualnych białych znaków, oraz sprowadzenie do małych liter. Następnie wypowiedziane zdanie zostaje rozbite według spacji, a jego elementy zostają przypisane do listy. W przypadku pojedynczego słowa zostaje ono w całości umieszczone jako lista jednoelementowa. W następnym kroku przebiegu metody następuje zagnieżdżenie, w którym odbywa się porównywanie elementów listy ze słownikiem kluczy zgodnie z miarą odległości Levenshteina^[11]. Służy ona do liczenia odległości edycyjnej. Jest to liczba działań potrzebna, aby przekształcić jeden wyraz w drugi. Działania wpływające na zmianę odległości to:

- wstawienie nowego znaku
- usunięcie znaku
- zamiana na inny znak

Po porównaniu danego elementu zostaje wybrany ten, który posiada najmniejszą wartość błędu. Następnie

następuje sprawdzenie, czy wartość błędu elementu odpowiada wcześniej zdefiniowanej oczekiwanej wartości granicznej. Jeżeli warunek zostaje spełniony, następuje po nim iteracja zmiennej prawidłowych odpowiedzi i kolejny obieg pętli. Po pełnym przebiegu, przez funkcję zwrócona zostaje wartość ilości poprawnych odpowiedzi. Dwie pozostałe funkcje odpowiedzialne za operacje na tekście różnią się od siebie przebiegiem działania oraz wartością zwracaną, jedna z nich odpowiada za przeprowadzenie porównań pełnego zdania bez zagnieżdżania w celu pozyskania pojedynczych słów, druga zaś zwraca wykryte słowo klucz, na którym później przeprowadzane są operacje.

3.2.2 Funkcja parsująca tekst oparta o jednostki wielowyrazowe

Tak samo jak funkcja zademonstrowana w 3.2.1, celem zademonstrowanej na poniższym rysunku 3.2.2 funkcji, jest operowanie na tekście otrzymanym po konwersji wypowiedzi użytkownika.

```
def get_event(input_text, interesting_list_word):
    deep_pavlov_prepro = ner_model([input_text])
    print(deep_pavlov_prepro)
    list_words = deep_pavlov_prepro[0][0]
    list_predict = deep_pavlov_prepro[1][0]
    list_word_predict = list(zip(list_words, list_predict))

    PERSON = ['B-PERSON', 'I-PERSON']
```

Rysunek 3.2.2 Część treści funkcji parsującej tekst używając modelu NER

Różni się ona, wykorzystując model NER^[12] służący do rozpoznawania wyrażeń, za pośrednictwem algorytmu wyszukiwania BERT. Dzięki wykorzystaniu metody rozpoznawania wyrażeń funkcja pozwala na znacznie obszerniejsze wychwytywanie klucza niż wcześniej zdefiniowany słownik. Dla przykładu posłużono się opisem imienia i nazwiska użytkownika. Określenie [PERSON] należy do zbioru wyrażeń wielowyrazowych mogących oznaczać imię oraz nazwisko użytkownika. Oznaczenia [B]-Beginning oraz [I]-Inside odpowiadają pierwszemu, oraz następującemu po nim dopasowaniu słów. Dzięki wykorzystaniu tej funkcji aplikacja pozwala na sprawdzenie poprawności i dopasowanie, imienia podanego przez użytkownika, daty jego urodzenia, oraz adresu zamieszkania. Pozostałe słowniki nie zostały zbudowane na podstawie tego typu wyrażenia z powodu trudności związanej z tematyką nauki modelu NER, oraz przez wymóg odpowiedniej treści kluczy potrzebny do prawidłowego przebiegu rozmowy.

3.2.3 Funkcje operujące na zwróconym kluczu

Funkcja przedstawiona na rysunku 3.2.3 jest jedną z wielu funkcji, na których oparty jest system przyznawania użytkownikowi punktów za prawidłowe odpowiedzi.


```

def get_experience(input_text, yes_no):
    global points
    stat, answer = parse_text_without_points(input_text, yes_no, 1)
    if stat == True and answer == "tak":
        points = points + 5
        return True, input_text
    elif stat == True and answer == "nie":
        return True, input_text
    else:
        return False, input_text

```

Rysunek 3.2.3 Treść przykładowej funkcji operującej na otrzymanym kluczu

Wywołuje ona wymaganą metodę operującą na otrzymanej wiadomości, przekazując jej treść wypowiedzi, słownik zawierający klucz, oraz maksymalną wartość różnicy w miary odległości między słowami. O ile wywołana metoda zakończy działanie pozytywnie, zwraca status ukończenia wykrywania klucza oraz sam klucz. Następnie funkcja operuje na otrzymanym kluczu, przyznając odpowiednie ilości punktów. Dla przykładu zademonstrowano prostą treść odpowiedzi na pytanie odnośnie posiadania doświadczenia w zawodzie. Funkcja ta, jak i jej podobne zwracają treść wypowiedzi lub klucza w celu utworzenia finalnego zapisu rozmowy. Oraz w zależności od odpowiedzi dodaje odpowiednią ilość punktów do zmiennej globalnej, w której przechowywany jest aktualny wynik użytkownika. W przypadku gdy status wywołanej metody wykrywania klucza jest negatywny, funkcja ta również otrzymuje wynik negatywny, wywołując powtórzenie pytania.

3.2.4 Funkcje operujące na wyrażeniach regularnych

Na podanym niżej przykładzie 3.2.4.a pokazana została funkcja sprawdzająca podany przez użytkownika numer telefonu komórkowego.

```

def check_phone_number(phone_number):
    if re.match('[0-9]{9}$', phone_number):
        return True
    else:
        return False

def get_phone_number(phone_number):
    type(phone_number)
    print(phone_number)
    phone_number = phone_number.replace(" ", "")
    if check_phone_number(phone_number):
        type(phone_number)
        return True, phone_number
    else:
        return False, phone_number

```

Rysunek 3.2.4.a Treść funkcji operującej na wyrażeniach regularnych

Funkcje wykorzystujące wyrażenia regularne różnią się od pozostałych w programie, nie mając do siebie przypisanego klucza odpowiedzi, nie przeprowadzają operacji wyszukiwania na tekście. Służą one do uniwersalnego porównania treści do wzorca. Powyższa funkcja sprowadza dowolną treść numeru do pojedynczej cyfry poprzez usunięcie białych znaków, następnie wywołując metodę operującą na wyrażeniu

regularnym. Sprawdza, czy w podanej treści znajduje się odpowiednia ilość znaków dla numeru telefonu komórkowego, oraz czy w jego skład wchodzi jedynie cyfry.

Na przykładzie 3.2.4.b Pokazany został fragment kolejnej funkcji używającej "Regex", odpowiedzialnej za sprawdzenie adresu e-mailowego użytkownika.

```
def check_email(email):  
    email = email.replace("małpa", "@")  
    email = email.replace("kropka", ".")  
    email = email.replace(" ", "")  
  
    regex = "^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+$)"  
    if re.search(regex, email):  
        print("Valid Email")  
        return True, email  
    else:  
        print("Invalid Email")  
        return False, email
```

Rysunek 3.2.4.b Treść drugiej funkcji operującej na wyrażeniach regularnych

Zostaje ona tutaj wspomniana, ponieważ jest szczególnym przypadkiem operowania na odebranej treści wypowiedzi. W funkcji następuje zamiana słów na znaki przed sprawdzeniem wyrażeniem regularnym. Jest to wywołane problematyczną kwestią niedoskonałości w zrozumieniu wypowiedzi. Słowa te nie zawsze są odbierane przez aplikacje jako symbole. Jest to prawdopodobnie najmniej prawidłowo zoptymalizowana funkcja, ponieważ występują liczne problemy ze zrozumieniem kilkuliterowych fraz oddzielonych kropką.

3.2.5 Funkcje Konwertujące treść wypowiedzi TTS i STT

Na poniższym rysunku 3.2.5 pokazany został fragment kodu. Zawiera się on w pliku odpowiedzialnym za przetwarzanie danych. Pokazana jest na nim funkcja przeprowadzająca konwersję wypowiedzi użytkownika na tekst. Współdziałała ona z funkcją, która jest odpowiedzialna za ustawienie działania nasłuchiwanie wypowiedzi użytkownika oraz jej zapis do pliku ".wav."

Na początku działania funkcji odbywa się nawiązanie połączenia z klientem Google API przeprowadzającym konwersję dźwięku, w celu wykorzystania metod, które zawiera. Następnie otwarty oraz odczytany przez aplikację zostaje wcześniej zapisany plik z treścią wypowiedzi użytkownika. Następuje sprawdzenie, czy treść istnieje, oraz czy po przeprowadzeniu transkrypcji spełnia błąd poprawności tekstu, jeżeli warunki te nie zostaną spełnione, aplikacja powtórzy proces zadawania pytania oraz nadpisze poprzednią wypowiedź. Po przejściu etapu sprawdzania wypowiedzi, funkcja zwraca przetworzoną treść w postaci tekstu, na którym można prowadzić dalsze operacje.

```

def stt_google_wav(audio_fname):
    print("Sending ", audio_fname)
    filename = audio_fname
    client = speech.SpeechClient()
    file_name = 'output.wav'
    with io.open(file_name, 'rb') as audio_file:
        content = audio_file.read()
        audio = types.RecognitionAudio(content=content)
    config = types.RecognitionConfig(
        encoding=enums.RecognitionConfig.AudioEncoding.LINEAR16,
        language_code='pl-pl')
    response = client.recognize(config, audio)
    if len(response.results) == 0:
        synthesize_text_with_audio_profile('Przepraszam, nic nie słyszę.')
    for result in response.results:
        if result == "":
            synthesize_text_with_audio_profile('Przepraszam, nic nie słyszę.')
            break
        print('Transcript: {}'.format(result.alternatives[0].transcript))
        if result.alternatives[0].confidence < 0.75:
            synthesize_text_with_audio_profile('nie zrozumiałam')
            break
        else:
            return result.alternatives[0].transcript
    os.remove(filename)
    return ""

```

Rysunek 3.2.5 Treść funkcji konwertującej treść wypowiedzi na tekst

Funkcją przeciwną do zademonstrowanej jest funkcja przeprowadzająca konwersję odwrotną ze wcześniej zdefiniowanej treści. W której następuje zbudowanie modelu rozmówcy opartego o język SSML^[13]. Po ustawieniu odpowiednich parametrów wywołana na modelu zostaje treść wiadomości tekstowej, którą odczytuje. Finalnie następuje tak jak w powyższej metodzie zapis treści do tymczasowego pliku ".wav"

3.3 Najważniejsze biblioteki i technologie

Aplikacja do działania wymaga spełnienia odpowiednich na ten moment warunków hardware, wnioskuje, że są one wywołane nieoptymalizowaniem pobranych bibliotek potrzebnych do działania uniwersalnie. Wymagane, poza podstawowym sprzętem komputerowym, są sterowniki Nvidia, z czym powiązana jest odpowiednia karta graficzna, poza tym dostęp do internetu w celu połączenia z Google cloud, oraz urządzenie odtwarzające dźwięk wraz z mikrofonem w celu prowadzenia rozmowy.

3.3.1 Biblioteka DeepPavlov^[14]

Biblioteka służąca do tworzenia chat-botów oraz wirtualnych asystentów, posiadająca możliwości przeprowadzania operacji na języku naturalnym. Zawiera wbudowane algorytmy oparte o uczenie maszynowe^[15], zbudowane korzystając z upublicznionych struktur. Posiada wbudowane metody pozwalające na przeprowadzanie operacji związanych z prowadzeniem konwersacji z aplikacją. Wykorzystana została do zaimplementowania algorytmu porównawczego modelu BERT oraz NER. Obsługa treści tekstowych oraz dźwiękowych została zastąpiona systemem Google API.

3.3.2 Moduł os^[16]

Moduł zapewniający możliwości operowania na systemie operacyjnym z poziomu języku Python. Niezależny od wykorzystywanego systemu operacyjnego, pozwala na pozyskiwanie informacji na temat systemu oraz wprowadzanie modyfikacji opartych o działanie na plikach. Wykorzystany w aplikacji w celu modyfikacji referencji systemowej związanej z autoryzacją połączenia z serwerem Google. Także użyty do operacji na utworzonych plikach

3.3.3 Pakiet stringdist^[17]

Pakiet umożliwiający sprawdzenie dopasowań po wykonanej konwersji dźwięku. Bazuje na wyliczaniu dystansu pomiędzy danymi a pożądanymi znakami, operując na wcześniej wspomnianej mierze odległości Levenshteina. Wykorzystując ten pakiet, zostało wprowadzone, odpowiednie dla każdej metody, dopasowanie słów wypowiedzianych przez użytkownika z kluczem. Poprzez wcześniejsze ustalenie marginesu błędu, który jest konieczny w przypadku potrzeby porównywania słów, których treść nie zawsze występuje w tej samej formie.

3.3.4 Moduł re^[18]

Moduł wprowadzający funkcjonalności oparte o wyrażenia regularne. Wyrażenia te symbolizują odpowiednie dla ich struktury zbiory treści tekstowej. Pozwalają na uniwersalizację sprawdzanej treści do danego wzorca zbudowanego, z których symboli znaczenie jest ściśle określone w ich dokumentacji. Wykorzystany został w aplikacji do operacji porównywania treści wypowiedzianego adresu e-mail użytkownika oraz numeru telefonu, z powodu dowolności, a zarazem ograniczeń związanych z ich treścią.

3.3.5 Klasa opakowująca Google maps^[19]

Klasa odpowiedzialna za dostarczenie aplikacji metod operujących na mapach Google. Jest ona tylko łącznikiem dostarczającym narzędzia, wymaga wcześniejszego zadeklarowania połączenia z chmurą Google. Pozwala na wprowadzenie geokodowania, czyli ustalenia współrzędnych geograficznych na podstawie danych związanych z istniejącym adresem.

3.3.6 Moduły pyaudio^[20], wave^[21], io^[22]

Moduły odpowiedzialne za przeprowadzanie operacji na plikach odpowiedniego typu. W aplikacji pyAudio służy do odsłuchu dźwięku oraz jego odczyt poprzez udostępnienie portu dźwiękowego systemu. Wave pozwala na przeprowadzanie operacji typu zapis i odczyt, na plikach o rozszerzeniu ".wav" zawierających dźwięk z zapisanego fragmentu rozmowy. Io wraz ze wcześniej wspomnianym modułem Os zawiera odpowiednie metody do prowadzenia operacji na plikach o dowolnie zadeklarowanym rozszerzeniu.

3.3.7 Implementacja Google cloud

Połączenie z chmurą za pomocą Pythona poprzez wcześniej wspomnianą metodę modyfikującą referencje, pozwala aplikacji na korzystanie z bibliotek i technologii wbudowanych w chmurę. Za jej pośrednictwem odbywa się korzystanie funkcji z API, text to speech, speech to text oraz Google Geocoding. Odpowiednio odpowiedzialnych za konwersje mowy na tekst zrozumiały dla systemu, tekstu na mowę, oraz operacji na podanym przez użytkownika adresie.

4. Testowanie aplikacji

W tym rozdziale opisany zostanie przebieg procesu testowania aplikacji wraz z napotkanymi w trakcie przeciwnościami. Aplikacja od samego początku rozpoczęcia pracy wywoływała problemy wymagające przeprowadzenia testów na różnego typu sprzęcie komputerowym. Głównym napotkanym na początku problemem był brak możliwości uruchomienia zaimplementowanych bibliotek na komputerze ze zintegrowaną kartą graficzną. Powodem napotkanego problemu był wymóg posiadania wcześniej wspomnianych sterowników Nvidia. Napotkany problem nie został rozwiązany, zamiast tego praca nad projektem została przeniesiona na komputer osobisty posiadający odpowiednie oprogramowanie. Błąd ten został skonsultowany z firmą D-labs, która posiadała aplikacje o podobnej funkcjonalności, lecz problem nie został przez nich napotkany, prawdopodobnym powodem jego występowanie było błędne zainstalowanie odpowiednich bibliotek, lub błąd w oprogramowaniu samego komputera.

Proces testowania aplikacji wykonywany był metodą testów użytkownika, z powodu braku doświadczenia w dziedzinie przeprowadzania testów jednostkowych na tego typu systemie. Poprzez skrupulatne sprawdzanie reakcji poszczególnych funkcji na warunki graniczne. Każdej funkcji operującej na słowniku kluczy przypisywane były treści puste, testując czy wystąpi odpowiednio zaprogramowana reakcja na brak wiadomości. Następnie wprowadzana została treść nieposiadająca żadnego związku z kluczem, oczekując braku zrozumienia wypowiedzi przez program, oraz wywołania powtórzenia danej funkcji. Finalnie podając treść prawidłową, odpowiednią dla występujących kluczy. Funkcjom operującym na treści wypowiedzi użytkownika przypisano etapy wyświetlania treści dla danego momentu, w celu przeprowadzenia nadzoru przebiegu następujących modyfikacji.

5. Podsumowanie

Celem niniejszej pracy było utworzenie aplikacji, używającej algorytmów wykorzystujących uczenie maszynowe potrzebne do stworzenia chat-bota przeprowadzającego rozmowę selekcyjną kandydatów na stanowisko menadżera. W skład aplikacji wchodzi podzespoły odpowiedzialne za konwersję dźwięku i tekstu, klucz zawierający dane uwierzytelniające Google, oraz główny plik aplikacji zawierający funkcję operującą na przetworzonej treści. Zgodnie z założeniami wymaganych przez aplikacje technologii została ona utworzona w języku Python. Ukończona aplikacja jest funkcjonalna i potwierdza idee, dzięki której można rozwiązać problem określony w celu pracy.

Proces tworzenia aplikacji był naprzemienny z nauką implementowanych systemów, co znacząco ograniczyło możliwości usprawnień nowo nabytymi umiejętnościami wcześniej utworzonych elementów. Korzystniejszym scenariuszem byłoby utworzenie dokładnego opisu aplikacji następnie, zagłębienie się w tematykę wykorzystywanych technologii. Co wpłynęłoby na uniwersalność aplikacji

Aplikacja została ukończona w stanie prototypowym. Zgodnie z obranymi wymaganiami, wszystkie potrzebne funkcjonalności zostały w niej umieszczone. Jest ona zdolna do operacji na plikach dźwiękowych, przeprowadzając konwersje do tekstu, do operowania na tekście przetwarzając go na wiadomości słyszane przez użytkownika. Posiada ona budowę logiczną, zgodną z przebiegiem rozmowy kwalifikacyjnej, której elementy zostały pozyskane w przeprowadzonym wywiadzie. Jest w stanie od początku do końca przeprowadzić około siedmiominutowy wywiad z użytkownikiem, zapisując otrzymane informacje oraz przyznawać mu punkty za odpowiedzi zgodnie ze zdefiniowanym kluczem. Wyeliminowano z niej znaczącą ilość błędów występujących w okresie testowania przez użytkownika.

W celu uzyskania w pełni pożądanego efektu wymaga wprowadzenia usprawnień dotyczących treści słowników. Jest to możliwe do wykonania poprzez rozbudowę ich zgodnie z modelem NER, tworząc segmenty złożone z wyrażen wielowyrazowych, lub umieszczenie większej ilości słów kluczy zgodnych treścią z zadawanym pytaniem. Innym elementem wymagającym usprawnienia w przypadku kontynuacji rozwoju oprogramowania byłaby logika aplikacji, której rozbudowa powinna zostać skierowana w bardziej responsywny kierunek. Większa ilość pytań powinna być generowana na podstawie odpowiedzi użytkownika. Część z nich zgodnie z taką logiką powinna zostać pomijana automatycznie. Przyznając brak punktów za pominięte pytania jeżeli były one istotne dla przebiegu rozmowy. Finalnym krokiem rozwoju powinno być wyeksportowanie aplikacji w celu umieszczenia jej na odpowiednim serwerze, zapewniając ostateczną funkcjonalność aplikacji, czyli możliwość przeprowadzania rozmów z aplikantami.

Wykaz literatury:

[1] Strona domowa firmy D-labs

[online]. [dostęp 16.09.2019] Dostępny w internecie:

<https://dlabs.ai/>

[2] Czym jest Hacathon

Hacathon [online]. Wikipedia: wolna encyklopedia. [dostęp], Dostępny w internecie:

<https://pl.wikipedia.org/wiki/Hackathon>

[3] Czym jest Chat-bot

Chat-bot [online]. Wikipedia: wolna encyklopedia. [dostęp], Dostępny w internecie:

<https://pl.wikipedia.org/wiki/Chatbot>

[4] Strona domowa Pythona

[online]. Dostępny w internecie:

<https://www.python.org/>

[5] Strona domowa Google API

[online]. Dostępny w internecie:

<https://developers.google.com/drive/?hl=pl>

Opis Cloud Speech API

Ravulavaru A. (2018) Google Cloud AI Services Quick Start Guide: Build intelligent applications with Google Cloud AI services. Cloud speech API. s. 144-149 Częściowo Dostępny w internecie:

<https://books.google.pl/books?id=aHteDwAAQBAJ&lpg=PA154&dq=alternatives%5B0%5D.transcript&hl=pl&pg=PA154#v=onepage&q&f=false>

[6] Strona domowa Microsoft Windows

[online]. Dostępny w internecie:

<https://www.microsoft.com/en-us/windows>

[7] Aplikacja wzorcowa firmy D-labs

[online]. [dostęp 16.09.2019] Dostępny w internecie:

<https://github.com/dlabsai/hackathon2019>

[8] Strona domowa pyCharm

[online]. Dostępny w internecie:

<https://www.jetbrains.com/pycharm/>

[9] Strona domowa Nvidia

[online]. Dostępny w internecie:

<https://www.nvidia.com/pl-pl/>

[10] Dokumentacja BERT dla DeepPavlov

BERT in DeepPavlov [Online]. [dostęp:07.10.2019] Dostępny w internecie:

<http://docs.deeppavlov.ai/en/master/features/models/bert.html>

[11] Algorytm Levenshteina

Odległość Levenshteina [online]. Wikipedia: wolna encyklopedia. [dostęp], Dostępny w internecie:

https://pl.wikipedia.org/wiki/Odleg%C5%82o%C5%9B%C4%87_Levenshteina

[12] Dokumentacja NER dla DeepPavlov

Named Entity Recognition (NER) [Online]. [dostęp:07.10.2019] Dostępny w internecie:

<http://docs.deeppavlov.ai/en/master/features/models/ner.html>

Opis modelu NER

Král P. (red.) Matoušek V. (red.) (2005), Text, Speech, and Dialogue, 18th International Conference, TSD 2015, Pilsen, Czech Republic, September 14-17, 2015, Proceedings. s. 61-70. Częściowo dostępny w internecie: <https://books.google.pl/books?id=SQuVCgAAQBAJ&lpg=PA62&dq=multi%20word%20entity%20B-person&hl=pl&pg=PA62#v=onepage&q&f=false>

[13] Dokumentacja SSML

[online]. Dostępny w internecie:

<https://developers.google.com/assistant/actions/reference/ssml>

[14] Strona domowa DeepPavlov

[online]. Dostępny w internecie:

<http://deeppavlov.ai/>

[15] Czym jest uczenie maszynowe

Uczenie maszynowe [online]. Wikipedia: wolna encyklopedia. [dostęp], Dostępny w internecie:

https://pl.wikipedia.org/wiki/Uczenie_maszynowe

[16] Dokumentacja OS

Miscellaneous operating system interfaces [Online]. [dostęp:08.10.2019] Dostępny w internecie:

<https://docs.python.org/3/library/os.html>

[17] Dokumentacja Stringdist

Bulkin O.: StrindDist 1.0.9 Documentation [Online]. [dostęp: 11.11.2019] Dostępny w internecie:

<https://pypi.org/project/StringDist/>

[18] Dokumentacja re

Regular expression operations [Online]. [dostęp:12.11.2019] Dostępny w internecie:
<https://docs.python.org/3/library/re.html>

[19] Repozytorium serwisu Google Maps Dla Pythona

[online]. Dostępny w internecie:

<https://github.com/googlemaps/google-maps-services-python>

[20] Dokumentacja pyAudio

Pham H.: pyAudio 0.2.11 Documentation [Online]. [dostęp: 11.11.2019] Dostępny w internecie:
<https://people.csail.mit.edu/hubert/pyaudio/docs/>

[21] Dokumentacja wav

Read and write WAV files [Online]. [dostęp:11.11.2019] Dostępny w internecie:
<https://docs.python.org/3/library/wave.html>

[22] Dokumentacja io

[online]. Dostępny w internecie:

<https://docs.python.org/3/library/io.html>

Dodatek A. Spis zawartości dołączonej płyty CD

- Praca inżynierska – niniejszy dokument.
- Kod źródłowy aplikacji zawarty w plikach o rozszerzeniu “.py”.
- Plik README zawierający opis poszczególnych elementów znajdujących się na płycie.
- Film demonstrujący przebieg działania aplikacji.