Wydział Informatyki i Telekomunikacji
Politechnika Krakowska

Raport z projektu: „Asystent Głosowy”

Przetwarzanie Języka Naturalnego
(Natural Language Processing)

Autorzy: R.K., M.K., G.L.

Kraków, Styczeń 2023

Abstrakt

Niniejszy raport zawiera opis realizacji projektu pt. „Asystent Głosowy” na przedmiocie „Przetwarzanie Języków Naturalnych”. Asystent Głosowy umożliwia korzystającemu z niego użytkownikowi na zadawanie pytań i uzyskiwanie odpowiedzi. Działanie programu odbywa się w formie prostej konwersacji, wykorzystując mechanizmy rozpoznawania mowy (zamiany mowy na tekst), analizy semantycznej podanego zapytania, oraz syntezy mowy (zamiany tekstu na mowę).

W kolejnych sekcjach tłumaczymy, jakie problemy musieliśmy rozwiązać i jak wykorzystaliśmy wiedzę pozyskaną na wykładach do przygotowania omawianej aplikacji. Opisujemy także działanie silnika naszego Asystenta Głosowego.

Spis treści

1. Wprowadzenie

Opisujemy tutaj problem jaki mieliśmy do rozwiązania w praktycznym projekcie opracowanym na przedmiocie „Przetwarzanie Języka Naturalnego”.

1.1. Cel projektu

Celem projektu było wykorzystanie nabytych na zajęciach wiadomości z tematu przetwarzania języka naturalnego i zbudowanie programu rozwiązującego zadany praktyczny problem.

1.2. Opis zadanego problemu

Naszym problemem do rozwiązania było skonstruowanie **Asystenta Głosowego** w języku programowania **`Python`**. Proponowanymi bibliotekami były „**`Blather`**” (*analiza głosu*), „**`Sphinhx`**” (*generowanie dokumentacji technicznej*) oraz „**`PyFestival`**” (system syntezy mowy). W projekcie mieliśmy uwzględnić system analizowania poleceń przy użyciu technik konstrukcji ChatBota, takich jak: wektory TF-IDF (ang. *term frequency, inverse document frequency*), wyszukiwanie semantyczne, ekstrakcja informacji przy użyciu wyrażeń regularnych.

1.3. Zastosowania Asystenta Głosowego

Skupiliśmy się na tym, aby Asystent Głosowy rozumiał i wykonywał kilka podstawowych poleceń, takich jak uruchamianie programów, odczytywanie daty i godziny, konwersacja o samopoczuciu.

2. Projektowanie Asystenta Głosowego

Omawiamy jakie narzędzia (spośród proponowanych) wykorzystaliśmy do realizacji praktycznego projektu, oraz jakie napotkaliśmy początkowo trudności.

2.1. Wybór bibliotek i narzędzi

Projekt został napisany w języku **`Python`** z wykorzystaniem IDE „**`Visual Studio Code`**”.

Zastosowaliśmy następujące biblioteki:

- „**`tkinter`**” – graficzny interfejs użytkownika
- „**`Python Image Library`**” – ładowanie obrazów z folderu projektu
- „**`Unidecode`**” – normalizacja zapytań użytkownika przed porównaniem ich z lokalną bazą akceptowanych zapytań
- „**`scikit-learn`**” – wektoryzacja TF-IDF akceptowanych (wpisanych w bazę danych) zapytań.
- „**`NumPy`**” – porównywanie zapytania użytkownika z matrycą TF-IDF.
- „**`re`**” – wyszukiwanie wyrażeń regularnych w zapytaniach użytkownika
- „**`SpeechRecognition`**” – zamian mowy na tekst (wymaga stałego połączenia z Internetem, wykorzystuje „***Google Cloud Speech API***”).
- „**`pyaudio`**” – odtwarzanie dźwięku w Python 3.
- „**`pyttsx3`**” – nakładka do systemowego (Windows, Linux) SAPI (Speech API).

Biblioteki, z których zrezygnowaliśmy podczas projektowania Asystenta Głosowego:

- „**`Blather`**”, „**`PyFestival`**” – występowały problemy z instalacją. Powodami niepowodzenia instalacji były: nowsza wersja języka Python (**`3.11`**), niepoprawnie napisane pliki `Makefile` w bibliotece „**`Festival`**” (uniemożliwiające kompilację natywnej biblioteki zarówno pod systemem Windows jak i systemem Ubuntu).

2.2. Docelowa architektura systemu

Projekt **Asystenta Głosowego** jest napisany w **Pythonie** (wersji 3), który jest językiem programowania **wysokiego poziomu**. Interpretery języka Python dostępne są na wielu architekturach systemowych. Skupiliśmy się na tym, aby możliwe było uruchomienie projektu na systemach „**Microsoft Windows**” (**`Windows 10`**) oraz „**GNU/Linux**” (**`Ubuntu`**).

Asystent Głosowy obsługuje polecenia **wyłącznie w języku polskim**.

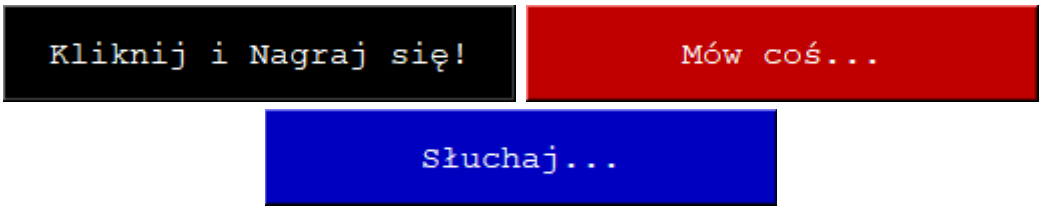
3. Implementacja Asystenta Głosowego

W tej sekcji opisujemy kolejne zagadnienia, na jakie musieliśmy zwrócić uwagę, oraz algorytmy które musieliśmy zaimplementować, żeby zbudować końcowy projekt.

3.1. Integracja szaty graficznej

W celu ułatwienia korzystania z Asystenta Głosowego, przygotowaliśmy prosty graficzny interfejs użytkownika. W oknie jest przycisk do nagrywania mowy, można przeczytać też polecenia do jakich ma zastosować się użytkownik („*Kliknij i Nagraj się*”, „*Mów coś...*” , „*Słuchaj...*”). Widoczne jest także domyślne urządzenie nagrywające dźwięk oraz domyślne urządzenie odtwarzające dźwięk.

Przyciski z biblioteki „**tkinter**”, jakie może zobaczyć użytkownik:



3.2. Ładowanie przykładowego zestawu danych

W folderze projektu zapisane są zewnętrzne dane w następujących plikach tekstowych:

- „**welcomes.txt**” – losowe powitania wymawiane przez Asystenta. Każde powitanie zapisane jest w osobnej linijce.
- „**backlash.txt**” – losowe odpowiedzi wymawiana przez Asystenta, kiedy zapytanie użytkownika nie pasuje do żadnego zestawu danych. Każda odpowiedź zapisana jest w osobnej linijce.
- „**dialogue.txt**” – zestawy scenariuszy (lista pytań oraz lista losowych odpowiedzi) obsługiwanych przez Asystenta Głosowego.

Scenariusze zapisane są w następującym formacie:

```
# to jest komentarz, komentarze są ignorowane
# przez interpreter scenariuszy.

[
    pytanie od użytkownika
    podobne pytanie, inaczej sformułowane
-
    pierwsza losowa odpowiedź
    druga losowa odpowiedź
    trzecia losowa odpowiedź
]

[
    pytanie od użytkownika, na które asystent musi zareagować
    podobnie sformułowane pytanie, np. z innym czasownikiem
-
    akcja do wykonania przez asystenta, np. podanie godziny
]

# otwarty nawias kwadratowy (otwarcie grupy),
# myślnik (oddzielenie pytań od odpowiedzi),
# zamknięty nawias kwadratowy (zamknięcie grupy).
```

3.3. Wektoryzacja pytań, dopasowywanie odpowiedzi

Słownik TF-IDF generowany jest poprzez „spłaszczenie” listy dwuwymiarowej (lista załadowanych grup pytań). Przykładowo, dla podanej bazy scenariuszy (z pliku „**`dialogue.txt`**”):

```
[
    co u ciebie
    co slychac
    jak sie masz
    jak sie czujesz
    jak tam
-
    odpowiedzi do pierwszej grupy pytań
]

[
    podaj godzine
    ktora godzina
    ktora mamy godzine
-
    odpowiedzi do drugiej grupy pytań
]
```

zostanie wygenerowany słownik z odpowiednimi identyfikatorami grup:

```
# słownik do testowania wektorów zapytań.
["co", "u", "ciebie", "slychac", "jak", "się", "masz",
"czujesz", "tam", "podaj", "godzine", "ktora", "godzina",
"mamy"]

# spłaszczona lista wszystkich pytań, wykorzystywana
# przy znajdywaniu największego cosinusa między wektorami.
["co u ciebie", „co slychac", "jak sie masz", "jak sie
czujesz", "jak tam", "podaj godzine", "ktora godzina", "ktora
mamy godzine"]

# lista indeksów dopasowanych grup odpowiedzi,
# jeśli któreś pytanie z listy pasuje do pytania
# zadanego przez użytkownika. W tym przykładzie
# pierwszy pięć pytań odpowiada zestawowi ID=0,
# a ostatnie trzy pytania odpowiadają zestawowi ID=1
[0, 0, 0, 0, 0, 1, 1, 1]
```

Dopasowywanie zapytania użytkownika (ang. query) do bazy pytań (w celu ustalenia odpowiedniej odpowiedzi przez Asystenta) odbywa się poprzez:

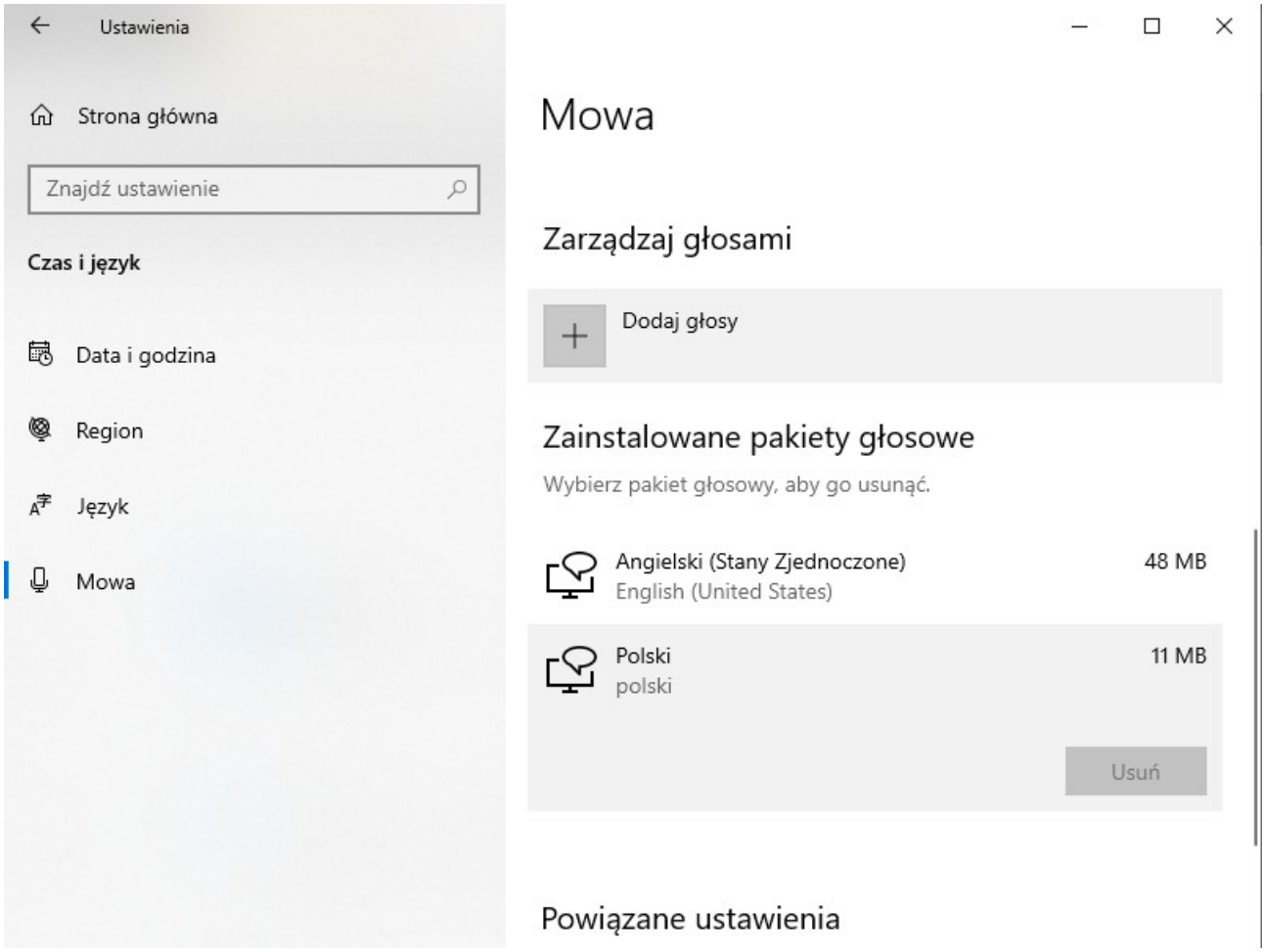
- jednokrotne przygotowanie matrycy słów z dostępnych pytań (spłaszczonej listy wszystkich dostępnych pytań w bazie) metodą „*`TfidfVectorizer::fit_transform(flat_questions)`*”;
- wektoryzację zapytania użytkownika względem przygotowanej matrycy słów (listy dostępnych pytań), stosując metodę „*`TfidfVectorizer::transform([query.lower()])`*”;
- dopasowanie najlepszego wektora z listy pytań do wektora zapytania użytkownika, wyszukując największy cosinus spośród wszystkich par wektorów wzorem: $\cos(\theta) = (\mathrm{A} \cdot \mathrm{B}) \div (|\mathrm{A}||\mathrm{B}|)$

3.4. Zamiana mowy na tekst (pytania użytkownika)

Zamiana tekstu na mowę odbywa się w metodzie „*`Speech::RecognizeText()`*” z wykorzystaniem biblioteki „*`speech recognition`*”. Próbka głosu (nagrywana maksymalnie 5 sekund) wysyłana jest do „Google Cloud” w celu szybkiego rozpoznania mowy i dopasowania transkryptu. Najlepszy transkrypt przekazywany jest do instancji Asystenta Głosowego (metoda „*`Chatting::findBestMatchForQuery(query)`*” i traktowany podobnie, jakby użytkownik napisał ręcznie wiadomość w oknie czatu z ChatBotem.

3.5. Integracja z silnikiem syntezy mowy (odpowiedzi Asystenta)

Asystent Głosowy obsługuje polecenia wyłącznie w **języku polskim**! System operacyjny zapewnia wbudowane **Speech API,** z którego korzysta biblioteka „`pyttsx3`”. Przykładowo, w systemie **Windows 10** wymagane jest zainstalowanie odpowiednich pakietów głosowych:



w przeciwnym wypadku synteza głosu odbędzie się w innym języku, lub wystąpi błąd, jeśli żaden pakiet głosowy nie został uprzednio zainstalowany.

Natomiast dla systemu operacyjnego **Ubuntu**, głosy do syntezy mowy dostarczone są poprzez bibliotekę „**`libespeak1`**”.

3.6. Integracja z usługami systemowymi

Wykorzystując wyrażenia regularne, Asystent Głosowy wyszukuje w tekście specjalne tokeny, takie jak np.:

- **`{TIME}`**, **`{DATOFWEEK}`**, **`{MONTH}`** – odpowiadają za wstawienie do tekstu odpowiedzi na podane pytanie, tj. podanie aktualnej godziny, podanie dnia tygodnia, podanie nazwy miesiąca.
- **`{ACTION:CLOSE}`** – Asystent zakończy działanie po odpowiedzi na pytanie (np. „*Proszę zamknąć program*” → „*Do widzenia!*”).
- **`{ACTION:START_WEBSITE}`** – Asystent włączy stronę internetową zaproponowaną przez użytkownika.

- **`{ACTION:START_PAINT}, {ACTION:START_NOTEPAD}`** – Asystent uruchomi wybrany program narzędziowy (*Notatnik, Microsoft Paint*).

4. Testowanie Asystenta Głosowego

Po zaimplementowaniu Asystenta Głosowego przystąpiliśmy do przetestowania poprawności działania programu.

4.1. Przygotowanie skryptów instalacyjnych

Aby ułatwić korzystanie z programu innym osobom, do projektu zostały dołączone skrypty wiersza poleceń (pliki „**`*.cmd`**” i „**`*.sh`**”) wykonujące automatycznie operacje:

- skrypt „**`prereq`**” – upewnia się, że w systemie zainstalowany jest interpreter języka **Python**, oraz doinstalowuje niezbędne biblioteki.
- skrypt „**`run`**” – przenosi kontekst powłoki do folderu z plikami projektu i uruchamia plik „**`main.py`**”.

4.2. Obsługa wybranych poleceń i scenariuszy

Asystent Głosowy obsługuje wyłącznie scenariusze przekazane w pliku tekstowym. Podczas wypisywania odpowiedzi do okna czatu, syntezowane są tylko odpowiedzi Asystenta, a pytania użytkownika są pomijane (gdyż użytkownik sam je zadał przez mikrofon).

W przypadku nierozpoznania któregoś z poleceń, Asystent Głosowy poprosi o powtórzenie pytania. Może się też zdarzyć, że Asystent będzie miał problem z rozpoznaniem niewyraźnej mowy, lub po prostu urządzenie nagrywające jest wyłączone w momencie nagrywania zapytania.

Asystent może zapytać użytkownika o samopoczucie, jest w stanie podać datę i godzinę, a także może uruchomić kilka programów narzędziowych lub witryn internetowych. Asystent nie jest niestety w stanie zintegrować się z kalendarzem systemowym, celem dodania jakiegoś przypomnienia lub nastawienia alarmu przypominającego w określony dzień o określonej godzinie.

5. Podsumowanie

Wykonanie projektu opisanego w zadanym problemie powiodło nam się. Wykorzystaliśmy odpowiednie techniki konstrukcji ChatBota. Przetestowaliśmy działanie programu na różnych przykładach użycia – Asystent Głosowy bardzo dobrze radzi sobie z rozpoznawaniem zaprogramowanych dla niego poleceń. Program potrafi wybierać odpowiednie odpowiedzi na zadawane pytania, oraz wykorzystuje silnik syntezy mowy do prowadzenia konwersacji z użytkownikiem.

Istnieje także możliwość rozbudowania takiego Asystenta w przyszłości celem dodania ciekawszych funkcjonalności, np. zwrócenia zapytań do wyszukiwarki internetowej, pobrania odpowiedzi z encyklopedii, czy rozpoznania symboli i wykonania prostych obliczeń matematycznych.

6. Źródła

- „**Dokumentacja biblioteki „Scikit Learn”**
  https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

- **Zagadnienia z wykładów (techniki budowania ChatBotów w języku programowania Python)**

- „**Python: Convert Speech to text and text to Speech**”
  Article Contributed By: „hameedkunkanoor”, „arorakashish0911”, ""sagartomar9927”.
  https://www.geeksforgeeks.org/python-convert-speech-to-text-and-text-to-speech/