Wykorzystanie narzędzi rozpoznawania mowy do sterowania podstawowymi funkcjami komputera

---

Piotr Nowakowski, Kacper Mróz, Kinga Płoszaj

Wstęp

Ta aplikacja jest narzędziem, które pozwala użytkownikom kontrolować klawiaturę i wybrane funkcje przeglądarki za pomocą poleceń głosowych. Aplikacja wykorzystuje bibliotekę przetwarzania języka naturalnego (NLP) Rasa oraz oprogramowanie do rozpoznawania głosu w Pythonie, aby interpretować i rozumieć polecenia wydane przez użytkownika.

Głównym celem projektu jest zapewnienie użytkownikom wygodnego i efektywnego sposobu interakcji z komputerem za pomocą głosu. Korzystając z tej aplikacji, użytkownicy mogą łatwo wykonywać zadania takie jak pisanie na klawiaturze, wyszukiwanie w Internecie i sterowanie odtwarzaczem multimedialnym bez konieczności używania tradycyjnej klawiatury lub myszy.

Oprócz biblioteki Rasa NLP, aplikacja wykorzystuje również oprogramowanie do rozpoznawania głosu, aby precyzyjnie interpretować wypowiadane przez użytkownika polecenia. Jest to możliwe dzięki zastosowaniu algorytmów uczenia maszynowego, które zostały wytrenowane na dużym zbiorze danych języka mówionego.

Cel projektu

Projekt ma na celu stworzenie aplikacji, która umożliwi sterowanie komputerem za pomocą mowy. Aplikacja będzie korzystać z biblioteki Rasa

NLP oraz oprogramowania do rozpoznawania głosu w języku Python. Będzie ona umożliwiać kontrolowanie funkcji klawiatury oraz niektórych funkcji przeglądarki internetowej za pomocą głosu.

Głównym celem projektu jest umożliwienie użytkownikom wygodnego i szybkiego sterowania komputerem za pomocą mowy. Dzięki zastosowaniu biblioteki Rasa NLP oraz oprogramowania do rozpoznawania mowy, aplikacja będzie w stanie rozpoznać różne polecenia i komendy wypowiadane przez użytkownika i odpowiednio na nie reagować.

Aplikacja będzie umożliwiać kontrolowanie niektórych funkcji przeglądarki internetowej, takich jak przewijanie strony, przełączanie między kartami czy wyszukiwanie zawartości. Będzie również obsługiwać komendy dotyczące kontroli klawiatury, takie jak kopiowanie i wklejanie tekstu czy zmiana języka klawiatury.

Cała aplikacja została napisana w języku Python, dzięki czemu będzie łatwa w integracji z innymi projektami oraz łatwa w rozszerzeniu o nowe funkcjonalności. Jest skierowana do osób, które chcą mieć możliwość szybkiego i wygodnego sterowania komputerem za pomocą mowy bez korzystania z klawiatury.

Wykorzystane Technologie

- Rasa - to otwartoźródłowa biblioteka przetwarzania języka naturalnego (NLP) do tworzenia chatbotów i asystentów głosowych. Zawiera ona ramę do tworzenia inteligentnych agentów konwersacyjnych,

potrafiących rozumieć i odpowiadać na wprowadzanie użytkownika w różnych kontekstach. Rasa zawiera narzędzia do tworzenia modeli zarządzania dialogiem, modeli rozumienia języka naturalnego oraz modeli syntezy mowy na podstawie tekstu.

- Google API do rozpoznawania mowy - Google API do rozpoznawania mowy to chmurowa usługa, która umożliwia deweloperom wykorzystanie zaawansowanej technologii rozpoznawania mowy Google'a w swoich aplikacjach. Może ona przepisywać mówione słowa na tekst, rozpoznawać polecenia mówione i odróżniać różnych mówców w rozmowie.
- gTTT (Google Text-to-Speech): gTTT (Google Text-to-Speech) to biblioteka syntezy mowy z tekstu, która umożliwia deweloperom generowanie mowy zapisanej w tekście. Korzysta ona z zaawansowanych algorytmów uczenia maszynowego Google'a, aby produkować wysokiej jakości, naturalnie brzmiącą mowę w różnych językach i akcentach.
- Kontrolery mikrofonu i głośników: Aby aplikacja mogła akceptować i odpowiadać na polecenia głosowe, musi mieć dostęp do mikrofonu i głośników użytkownika. Aplikacja zawiera kontrolery tych urządzeń, dzięki czemu może nagrywać i odtwarzać dźwięk według potrzeb.

Organizacja Działania

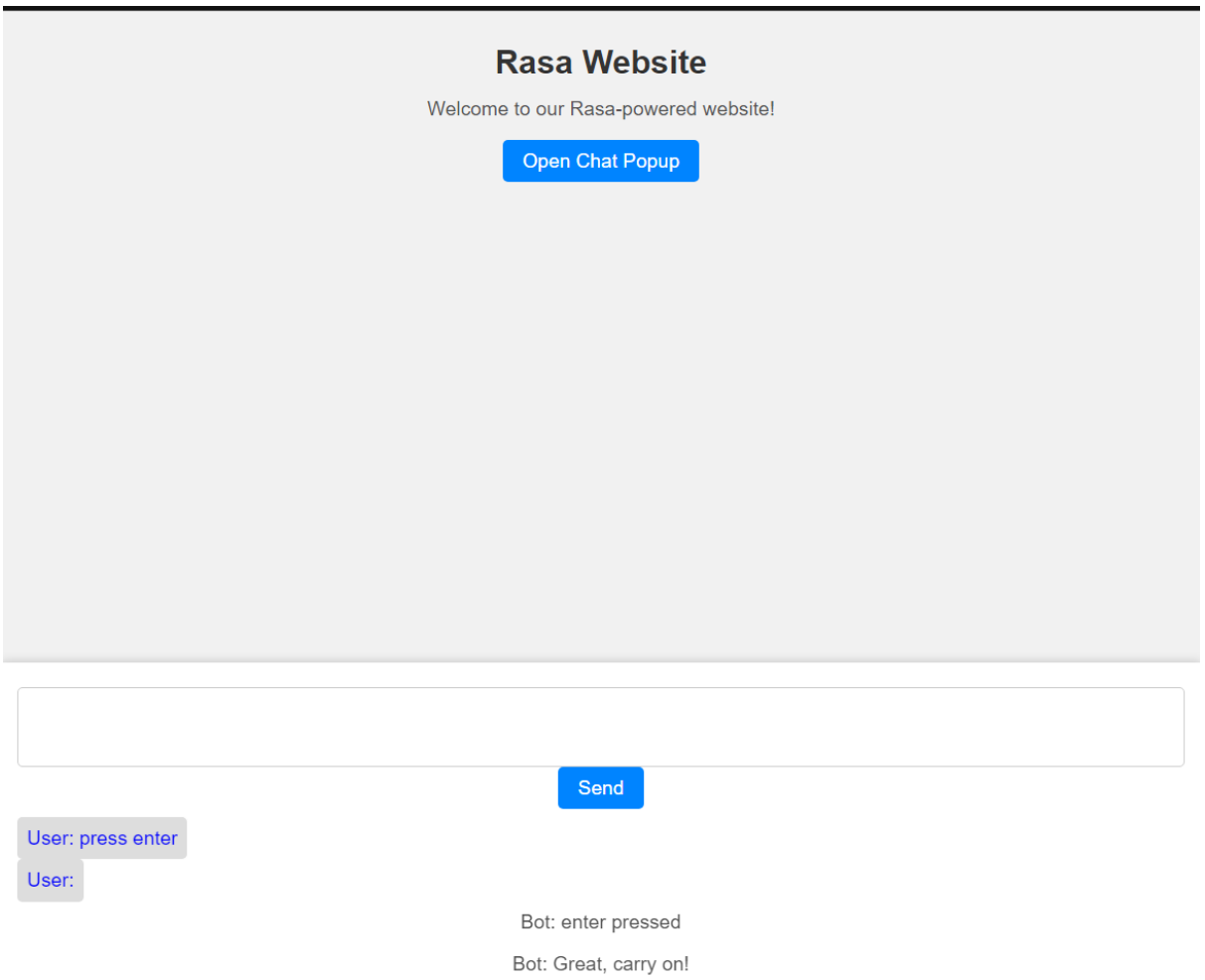
Ciąg interakcji z perspektywy użytkownika i aplikacji wygląda następująco:

- uruchomienie aplikacji - w obecnym stanie jest to osobne uruchomienie dwóch komponentów: serwera rasy do przetwarzania języka naturalnego i interakcji z systemem komputerowym, systemu do zapisu głosu z mikrofonu translacji na tekst i odtwarzaniu komunikatów zwrotnych z systemu rasa

- system odtwarza użytkownikowi informację o rozpoczęciu działania i nasłuchu komendy
- system testuje czy jest to mowa czy jedynie dźwięki otoczenia
- wykryta mowa jest wysyłana do API Googla który zwraca tekst wykryty w nagraniu
- przesłanie tekstu do serwera rozpoznania języka naturalnego w celu analizy
- Rasa tokenizuje, oznacza znajdujące się w zdaniu części mowy, dokonuje rozpoznania składników zdania (named entity recognition - NER) i tak preprocesowany tekst porównuje z wektorami dostarczonych danych treningowych. Następuje wyznaczenie najlepszego dopasowania i uruchomienie odpowieniej zdefiniowanej ,,akcji”
- wykonana jest akcja (lub jej brak jeśli nie została rozpoznana przez Rasę) w postaci wciśnięcia przycisku lub zadania w przeglądarce a do systemu generowania mowy z tekstu wysłana odpowiedź
- Aplikacja generuje plik mp3 z dostarczonej przez Rasę odpowiedzi używając biblioteki gTTT i odwtarza użytkownikowi

Dane wejściowe i ich przetwarzanie

W tym kontekście dane wejściowe to dane, które są wprowadzane do systemu, a w tym przypadku mowa. Te dane są przechwytywane za pomocą mikrofonu i oprogramowania rozpoznawania mowy, które konwertuje mówione słowa na tekst.
Aby lepiej zwizualizować to jak odbywa się przetwarzanie danych stworzyłem stronę internetową pozwalającą wyświetlać co w danym momencie wykonuje i zwraca w postaci tekstowej serwer Rasy.



Po wprowadzeniu hasła przez użytkownika dane wysyłane są do serwera Rasy pretrenowanego na dostarczonych przez nas danych.

```
2022-12-20 22:40:56 DEBUG    rasa.engine.graph  - Node 'select_prediction' running 'DefaultPolicyPredictionEnsemble.combine_predictions_from_kwargs'.
2022-12-20 22:40:56 DEBUG    rasa.core.policies.ensemble  - Made prediction using user intent.
2022-12-20 22:40:56 DEBUG    rasa.core.policies.ensemble  - Added `DefinePrevUserUtteredFeaturization(False)` event.
2022-12-20 22:40:56 DEBUG    rasa.core.policies.ensemble  - Predicted next action using MemoizationPolicy.
2022-12-20 22:40:56 DEBUG    rasa.core.processor  - Predicted next action 'action_enter' with confidence 1.00.
2022-12-20 22:40:56 DEBUG    rasa.core.actions.action  - Calling action endpoint to run action 'action_enter'.
2022-12-20 22:40:58 DEBUG    rasa.core.processor  - Policy prediction ended with events '[<rasa.shared.core.events.DefinePrevUserUtteredFeaturization object at 0x0000025077334310>]'.
2022-12-20 22:40:58 DEBUG    rasa.core.processor  - Action 'action_enter' ended with events '[BotUttered('enter pressed', {"elements": null, "quick_replies": null, "buttons": null, "attachment": null, "image"
: null, "custom": null}, {}, 1671572458.826133)]'.
2022-12-20 22:40:58 DEBUG    rasa.core.processor  - Current slot values:
        youtube_keyword: None
        session_started_metadata: None
2022-12-20 22:40:58 DEBUG    rasa.engine.runner.dask  - Running graph with inputs: {'__tracker__': <rasa.shared.core.trackers.DialogueStateTracker object at 0x00000250772FF310>}, targets: ['select_prediction'
] and ExecutionContext(model_id='ea9a54a4a68d48bd9f2340d8a1c69d42', should_add_diagnostic_data=False, is_finetuning=False, node_name=None).
2022-12-20 22:40:58 DEBUG    rasa.engine.graph  - Node 'rule_only_data_provider' running 'RuleOnlyDataProvider.provide'.
2022-12-20 22:40:58 DEBUG    rasa.engine.graph  - Node 'domain_provider' running 'DomainProvider.provide_inference'.
2022-12-20 22:40:58 DEBUG    rasa.engine.graph  - Node 'run_MemoizationPolicy0' running 'MemoizationPolicy.predict_action_probabilities'.
2022-12-20 22:40:58 DEBUG    rasa.core.policies.memoization  - Current tracker state:
[state 1] user intent: enter | previous action name: action_listen
[state 2] user intent: enter | previous action name: action_enter
```

Tu widzimy wynik przetwarzanie tego requestu przez serwer Rasy oras kolejne kroki prowadzące do predykcji action_enter z pełnym prawdopodobieństwem. Poniżej opiszę cały proces krok po kroku

Po skonwertowaniu danych wejściowych na tekst, następnie są przetwarzane przez moduł Natural Language Processing (NLP) Rasa w celu wyodrębnienia istotnych informacji i określenia odpowiedniej akcji do wykonania.Obejmuje to:

- Przetwarzanie wstępne: Dane wejściowe są czyszczone i wstępnie przetwarzane w celu przygotowania ich do dalszego przetwarzania. Obejmuje to usunięcie zbędnych słów lub fraz, standaryzację tekstu lub wykonanie innych rodzajów manipulacji danymi.
- Klasyfikacja intencji: Dane wejściowe są wprowadzane do modułu NLP, który wykorzystuje algorytmy uczenia maszynowego do klasyfikacji intencji stojących za tekstem. Może to obejmować identyfikację określonych słów kluczowych lub fraz i mapowanie ich do predefiniowanych intencji.
- Ekstrakcja encji: Moduł NLP wyodrębnia również odpowiednie podmioty z danych wejściowych, takie jak konkretne obiekty lub działania wymienione w tekście.
- Predykcja akcji: Na podstawie sklasyfikowanej intencji i wyodrębnionych encji, moduł NLP przewiduje odpowiednią akcję do podjęcia.
- Wykonanie akcji: Moduł akcji Rasa jest odpowiedzialny za wykonanie przewidywanej akcji. Może to obejmować interakcję z innymi systemami lub interfejsami API, sterowanie funkcjami klawiatury i przeglądarki internetowej lub wykonywanie innych zadań.
- Informacje zwrotne i aktualizacje: Rama Rasa wykorzystuje informacje zwrotne z wykonania akcji, aby zaktualizować i poprawić swoje zrozumienie danych wejściowych i odpowiednich działań, które należy podjąć. Może to obejmować aktualizację modeli uczenia maszynowego używanych przez moduł NLP lub dostosowanie mapowania intencji i działań.

Ważne jest, aby pamiętać, że dane wejściowe mogą wymagać przetworzenia lub oczyszczenia przed wprowadzeniem ich do modułu NLP. Może to obejmować usuwanie niepotrzebnych słów lub fraz, standaryzację tekstu lub wykonywanie innych rodzajów manipulacji danymi w celu zapewnienia, że mogą one być dokładnie przetwarzane przez moduł NLP.

Osobne przetwarzanie odbywa się jeśli chodzi o dane głosowe.
To realizowane jest poprzez dwie podstawowe klasy:

- recognize_speech_from_mic - klasa używająca biblioteki speech recognizer. Najpierw następuję wykrycie domyślnego dla systemu windows mikrofonu. Jeśli jest on dostępny dostosowany jest poziom na

którym przechwytywany jest dźwięk biorąc pod uwagę głos otoczenia (minimalizacji szumów i maksymalizacja dokładności). Na końcu, w momencie uzyskania satysfakcjonującego pliku audio następuje wysłanie go do api Google w celu otrzymaniu transkrypcji tekstu. Testowane były rozwiązania open-sourcowe np te dostępne w bibliotece speech_recognition jednak ich wyniki były niesatysfakcjonujące

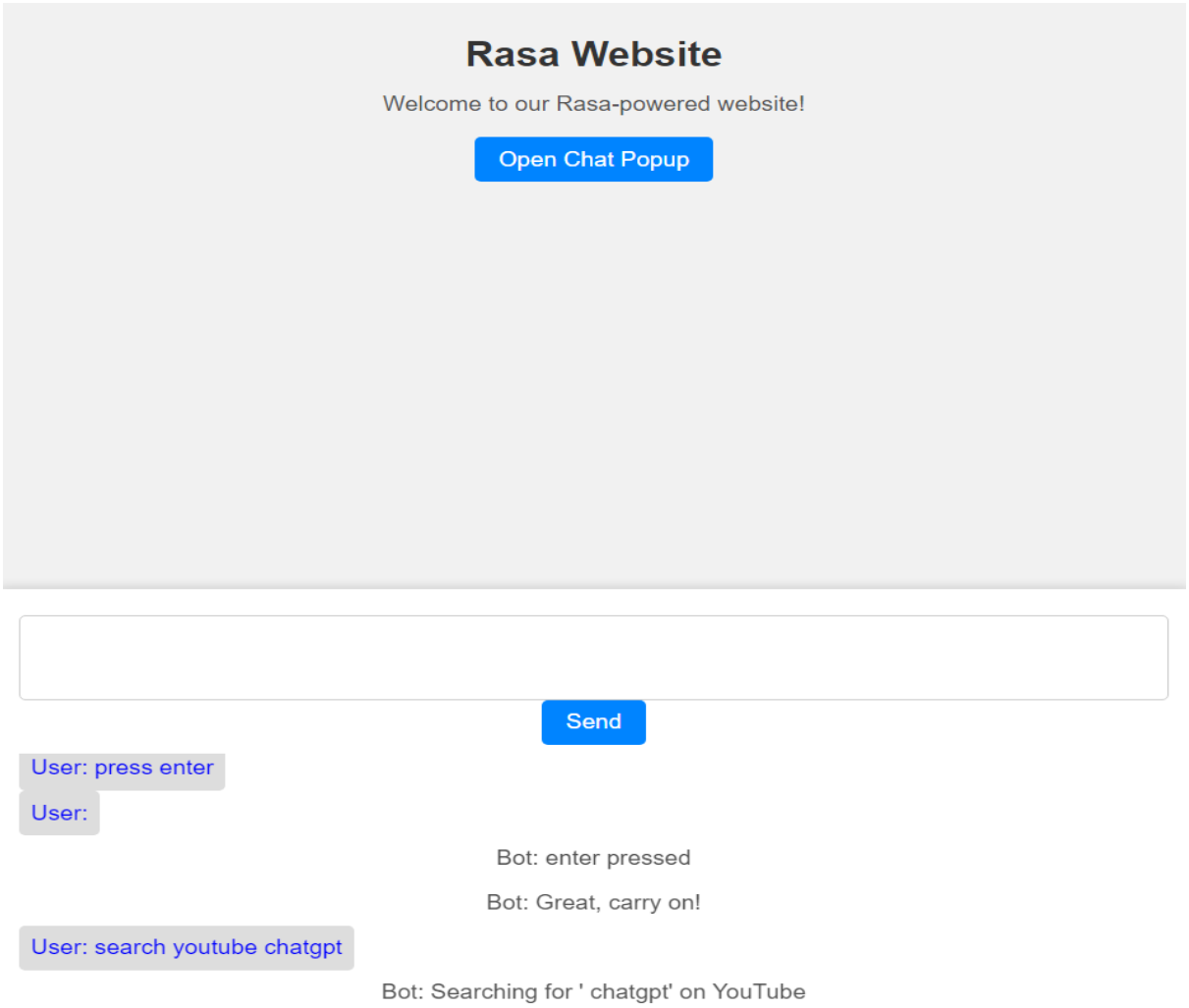
- read_to_user - dane otrzymane od rasy lub transkrypcje otrzymane z API za pomocą biblioteki gTTS (Google text to speech) są konwertowane do pliku audio mp3 następnie odtwarzanego przez głośniki dzięki funkcji playsound

Realizacja projektu

Projekt po uruchomienia obu serwerów jest zdolny do przetwarzania mowy użytkowniak w tekst, analizy jego intencji i odpowiedzi na dane potrzeby zgodnie z wcześniej zapisanymi przez programistę warunkami.
Zwizualizuję całkowitą drogę użycia dla sytuacji w którego użytkownik chce wyszukać głosowo film na YouTube.
System wyszukuje komendę podaną zaraz po słowie kluczowym ,,YouTube”

```
(speech) C:\Users\pites2\Desktop\Inzynierka\voice_recognition>python main.py
<speech_recognition.Microphone object at 0x000001A7B9E5B550>
You can give me an instruction on what to do on your computer
Guess 1. Speak!
it's okey
You said: search YouTube chat
Guess 2. Speak!
```

Możliwa jest również analogiczna interakcja przez stronę



Wynikiem przedstawionych działań jest otwarcie okna przeglądarki z wyszukiwaniem podanego przez nas słowa

YouTube PL

chatgpt

Główna

Shorts

Subskrypcje

Biblioteka

Filtry

O tych wynikach

Czy sztuczna inteligencja zabierze nam pracę? Testuję ChatGPT!

11 tys. wyświetleń • 10 dni temu

Jak nauczyć się programowania

Kontakt: kamil@jaknauczycsieprogramowania.pl Artykuły, o których wspominam w filmie: ...

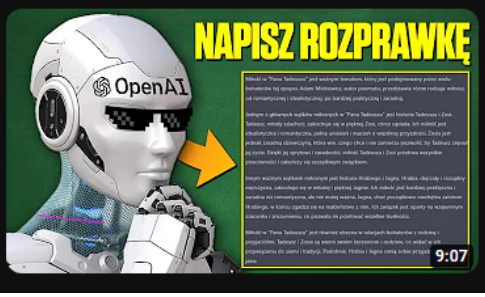
4K

ChatGPT - Twój asystent do pisania kodu i tekstów

16 tys. wyświetleń • 2 tygodnie temu

UW-TEAM.org

Sztuczna inteligencja dynamicznie się rozwija i potrafi już nie tylko pisać teksty, ale także generować działające kody źródłowe ...



Sztuczna inteligencja zrobi ci zadanie domowe w 10 sekund (Chat GPT)

25 tys. wyświetleń • 13 dni temu

Matura z Lewusem

Tutaj to jest https://chat.openai.com/ Jak chcesz wesprzeć to tutaj streamuje i możesz zostawić Twitch Prime za darmo: ...

Perspektywy rozwoju

Istnieje wiele dodatkowych funkcjonalności, które mogłyby zostać dodane do aplikacji wykorzystującej API Google do rozpoznawania mowy i framework Rasa do jego analizy, aby poprawić doświadczenie użytkownika i zwiększyć jej użyteczność. Niektóre z potencjalnych opcji mogą obejmować:

- Integracja z innymi platformami - aplikacja mogłaby zostać zintegrowana z innymi platformami lub usługami, takimi jak platformy społecznościowe lub streamingowe, aby umożliwić użytkownikom dostęp do tych usług i ich kontrolę za pomocą poleceń głosowych.
- Opcje personalizacji - aplikacja mogłaby oferować spersonalizowane rekomendacje i sugestie na podstawie poprzednich wyszukiwań i preferencji użytkownika, wykorzystując algorytmy uczenia maszynowego do dostosowywania ich do użytkownika.
- Wsparcie wielojęzyczne - oprócz obsługi języka polskiego, aplikacja mogłaby zostać rozszerzona o obsługę innych języków, umożliwiając użytkownikom przełączanie się między językami w razie potrzeby.
- Kontrola głosowa innych urządzeń - aplikacja mogłaby zostać zintegrowana z urządzeniami inteligentnymi - jak Google Home lub innymi podłączonymi do sieci urządzeniami, umożliwiając użytkownikom ich kontrolowanie za pomocą poleceń głosowych.
- Integracja z innymi asystentami AI - aplikacja mogłaby zostać zintegrowana z innymi asystentami AI, takimi jak Google Assistant lub Amazon Alexa, umożliwiając użytkownikom dostęp do funkcji i możliwości tych asystentów za pośrednictwem aplikacji.

Literatura

Google Cloud Speech-to-Text API: Quickstart Guide. Retrieved from https://cloud.google.com/speech-to-text/docs/quickstart-client-libraries

Rasa Open Source: Documentation. https://rasa.com/docs/

Budzianowski, P., & Casanueva, I. (2018). Building Chatbots with Python: Using Natural Language Processing and Machine Learning. Packt Publishing.

Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.

Raschka, S. (2015). Python Machine Learning: Unlock Deeper Insights into Machine Learning with This Vital Guide to Cutting-Edge Predictive Analytics. Packt Publishing.

Bocklisch, C., Ziebart, B., & Braune, C. (2017). Rasa: Open Source Language Understanding for Chatbots. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1,

Hulusic, V., & McBurney, P. (2017). A Survey of Chatbot Design Tools and Frameworks. In Proceedings of the 2nd International Conference on Human-Agent Interaction (pp. 3-11). Osaka, Japan: ACM.