POLITECHNIKA BYDGOSKA
im. Jana i Jędrzeja Śniadeckich

WYDZIAŁ TELEKOMUNIKACJI, INFORMATYKI
I ELEKTROTECHNIKI

Zaawansowane systemy multimedialne

Sprawozdanie

Temat: Aplikacja Android JetPack z wykorzystaniem biblioteki Google ML Kit do tłumaczenia tekstów

Wykonali:
Konrad Wierzbowski
Nr albumu: 112746

Anna Sarnecka
Nr albumu: 112740

Kajetan Romanowski
Nr albumu: 104760

Kierunek: Informatyka stosowa II stopień

1. Wstęp

Celem ćwiczenia było utworzenie translatora, który przetłumaczy słowa z dowolnych języków na inny dowolny język. Translator zostanie stworzony jako aplikacja mobilna zapisana za pomocą kodu języka Kotlin.

1.1 Cel i właściwości aplikacji

Aplikacje tłumaczące tekst z jednego języka na inny odgrywają kluczową rolę w dzisiejszym zglobalizowanym świecie. Wspomagają one komunikację między użytkownikami różnych kultur oraz ułatwiają dostęp do informacji napisanych w obcych językach. Jednym z języków programowania, który zyskuje na popularności w tworzeniu takich aplikacji, jest Kotlin. Kotlin, rozwijany przez JetBrains, łączy zalety języków programowania obiektowego i funkcyjnego, co czyni go idealnym narzędziem do tworzenia nowoczesnych, efektywnych i bezpiecznych aplikacji.

Właściwości Kotlin

Kotlin wyróżnia się wieloma cechami, które przyczyniają się do jego rosnącej popularności:

- **Bezpieczeństwo typów**: Kotlin zapewnia ścisłą kontrolę nad typami zmiennych, co zmniejsza ryzyko błędów związanych z nieprawidłowym przypisaniem wartości.
- **Współpraca z JVM**: Kotlin jest w pełni kompatybilny z Java Virtual Machine (JVM), co umożliwia wykorzystanie istniejących bibliotek Javy.
- **Nowoczesna składnia**: Składnia Kotlina jest przejrzysta i zwięzła, co zwiększa produktywność programistów.
- **Wysoka wydajność**: Kotlin umożliwia tworzenie aplikacji o wysokiej wydajności dzięki zaawansowanym mechanizmom optymalizacji kodu.

Aplikacja Symulująca Translator:

Celem aplikacji symulującej translator jest przetwarzanie tekstu wejściowego w jednym języku i generowanie odpowiadającego mu tłumaczenia w innym języku. Proces ten obejmuje kilka kluczowych kroków:

1. **Analiza tekstu źródłowego**: Rozbiór gramatyczny i leksykalny tekstu wejściowego.
2. **Mapowanie znaczeń**: Określenie odpowiednich słów i fraz w języku docelowym.
3. **Generowanie tekstu docelowego**: Konstrukcja poprawnego gramatycznie i semantycznie tekstu w języku docelowym.

W aplikacji wykorzystano następujące technologie:

- **Kotlin**: Do implementacji logiki aplikacji.
- **Biblioteka ML Kit – pakiet translate**

1.2 Biblioteka ML Kit

ML Kit to wszechstronna biblioteka stworzona przez Google, która oferuje narzędzia do implementacji funkcji związanych z uczeniem maszynowym na urządzeniach mobilnych. Jest częścią pakietu Firebase i umożliwia łatwe dodawanie funkcji AI do aplikacji na Androida i iOS bez konieczności posiadania głębokiej wiedzy na temat uczenia maszynowego. Oto kilka kluczowych cech i funkcji ML Kit:

Kluczowe Funkcje ML Kit

Rozpoznawanie Tekstu (Text Recognition):
- Wykrywanie i rozpoznawanie tekstu w obrazach.
- Obsługuje różne języki i style pisma.

Rozpoznawanie Obrazów (Image Labeling):
- Automatyczne identyfikowanie obiektów na zdjęciach.
- Obsługuje wstępnie wytrenowane modele Google oraz możliwość korzystania z własnych modeli.

Skanowanie Kodów Kreskowych (Barcode Scanning):
- Wykrywanie i rozpoznawanie różnych typów kodów kreskowych i QR.
- Szybkie i dokładne skanowanie w czasie rzeczywistym.

Rozpoznawanie Twarzy (Face Detection):
- Wykrywanie twarzy w obrazach oraz identyfikowanie cech takich jak oczy, nos, usta.
- Możliwość wykrywania uśmiechu, kierunku spojrzenia itp.

Śledzenie Pozycji i Ocenianie (Pose Detection and Estimation):
- Śledzenie położenia ciała i jego części na podstawie obrazu.
- Używane w aplikacjach sportowych, zdrowotnych i rozrywkowych.

Tłumaczenie Tekstu (Text Translation):
- Tłumaczenie tekstu między różnymi językami w czasie rzeczywistym.
- Obsługuje wiele popularnych języków.

Rozpoznawanie Punktów Orientacyjnych (Landmark Recognition):
- Identyfikowanie znanych miejsc i obiektów na świecie na podstawie zdjęć.

Zastosowania

ML Kit jest szeroko wykorzystywany w wielu aplikacjach mobilnych do:

- Skanowania dokumentów i kodów QR.
- Rozpoznawania produktów na zdjęciach i dostarczania dodatkowych informacji.
- Tworzenia aplikacji AR/VR z funkcjami śledzenia ruchu i rozpoznawania obiektów.
- Tłumaczenia tekstu na żywo, co jest przydatne w podróżach i komunikacji międzykulturowej.
- Analizowania treści multimedialnych, takich jak obrazy i wideo, w celu automatycznego kategoryzowania i tagowania.

Integracja i Użycie

ML Kit jest dostępny w ramach pakietu Firebase i można go łatwo zintegrować z aplikacjami mobilnymi za pomocą prostych SDK. Obsługuje zarówno platformy Android, jak i iOS, oferując przy tym możliwość pracy offline dla niektórych funkcji, co jest szczególnie ważne dla użytkowników z ograniczonym dostępem do internetu.

1.3 ML Kit Translate API

ML Kit's Translate API to narzędzie oferowane przez Google, które umożliwia łatwe dodawanie funkcji tłumaczenia tekstu do aplikacji mobilnych. Dzięki niej, aplikacje mogą tłumaczyć tekst między różnymi językami w czasie rzeczywistym, co jest przydatne w wielu kontekstach, takich jak podróże, nauka języków, komunikacja międzykulturowa i wiele innych.

Kluczowe Funkcje Translate API w ML Kit

Obsługa Wielu Języków:
- Translate API obsługuje ponad 50 języków, co pozwala na tłumaczenie tekstu między różnymi kombinacjami językowymi.

Tłumaczenie Offline:
- Jedną z najważniejszych funkcji jest możliwość tłumaczenia offline. Użytkownicy mogą pobrać pakiety językowe na swoje urządzenia, co pozwala na tłumaczenie bez potrzeby połączenia z internetem.
- Jest to szczególnie przydatne w sytuacjach, gdy dostęp do internetu jest ograniczony lub kosztowny.

Łatwa Integracja:

- Integracja Translate API jest prosta dzięki dostarczonym przez Google SDK. Programiści mogą szybko dodać funkcję tłumaczenia do swoich aplikacji bez konieczności głębokiego zrozumienia złożonych algorytmów tłumaczeniowych.
- Dokumentacja Google oferuje szczegółowe instrukcje oraz przykłady kodu.

Wysoka Dokładność i Wydajność:

- Translate API wykorzystuje zaawansowane modele tłumaczenia neuronowego, co zapewnia wysoką dokładność i naturalność tłumaczeń.
- Modele te są stale aktualizowane i ulepszane przez Google, co gwarantuje, że użytkownicy otrzymują najlepszą możliwą jakość tłumaczeń.

Personalizacja:

- Użytkownicy mogą dostosować funkcje tłumaczenia do swoich potrzeb, na przykład poprzez wybór preferowanych języków do tłumaczenia offline.

Zastosowanie Translate API

Translate API może być wykorzystywane w wielu różnych kontekstach, takich jak:

- **Aplikacje Turystyczne:**
  - Tłumaczenie znaków, menu, instrukcji itp. dla podróżników.
- **Komunikacja:**
  - Pomoc w komunikacji między osobami mówiącymi różnymi językami.
- **Edukacja:**
  - Pomoc uczniom i studentom w nauce nowych języków poprzez tłumaczenie tekstów.
- **Aplikacje Biznesowe:**
  - Tłumaczenie dokumentów, maili i innych treści biznesowych w czasie rzeczywistym.

2. Wykonanie ćwiczenia

Działanie aplikacji dzieli się na cztery etapy:

1. Wczytanie listy dostępnych języków
2. Sprawdzenie danych
3. Pobranie modelu danych
4. Tłumaczenie tekstu

Biblioteka ML Kit pozwala na prostą implementację tłumaczenia z użyciem wielu dostępnych języków. Poniższa funkcja jest odpowiedzialna za pobranie listy dostępnych języków do tłumaczenia. Metoda TranslateLanguage.getAllLanguages() pobiera listę dostępnych języków, następnie tworzy listę modeli języków.

```
private fun loadAvailableLanguages(){

    languageArrayList = ArrayList()

    val languageCodeList = TranslateLanguage.getAllLanguages()

    for (languageCode in languageCodeList){
        val languageTitle = Locale(languageCode).displayLanguage

        Log.d(TAG,  msg: "loadAvailableLanguages: languageCode: $languageCode")
        Log.d(TAG,  msg: "loadAvailableLanguages: languageTitle: $languageTitle")

        val modelLanguage = ModelLanguage(languageCode, languageTitle)

        languageArrayList!!.add(modelLanguage)
    }
}
```

Rys.1 - Implementacja ML Kit

Walidacja danych obejmuje tylko sprawdzenie czy tekst źródłowy nie jest pusty. Jeśli nie wprowadzono tekstu zostanie wyświetlony komunikat, aby wprowadzić tekst. Komunikat wyświetla się na kontrolce typu ***Toast***. Jeśli został wprowadzony tekst zostanie rozpoczęta translacja.

```
private fun validateData(selectedSourceCode:String,selectedTargetCode:Str

    if (inputData.isEmpty()){
        showToast("Enter text to translate...")
    }
    else{
        startTranslation(selectedSourceCode,selectedTargetCode,inputData,
    }

}
```

Rys.2 - Walidacja danych

Rozpoczęcie tłumaczenia - po rozpoczęciu procesu tłumaczenia użytkownikowi zostanie wyświetlone okienko ładowania, zostają ustawione opcje translatora (źródłowy i docelowy język), utworzony zostaje klient, następnie ustawione zostają warunki pobrania modelu języka (połączenie wi-fi).

```
private fun startTranslation(selectedSourceCode:String,selectedTargetCode:String,
    dialogMessage("Processing language model...")
    showDialog(true)

    translatorOptions = TranslatorOptions.Builder()
        .setSourceLanguage(selectedSourceCode)
        .setTargetLanguage(selectedTargetCode)
        .build()
    translator = Translation.getClient(translatorOptions)

    val downloadConditions = DownloadConditions.Builder()
        .requireWifi()
        .build()
```

Rys.3 - implementacja tłumaczenia

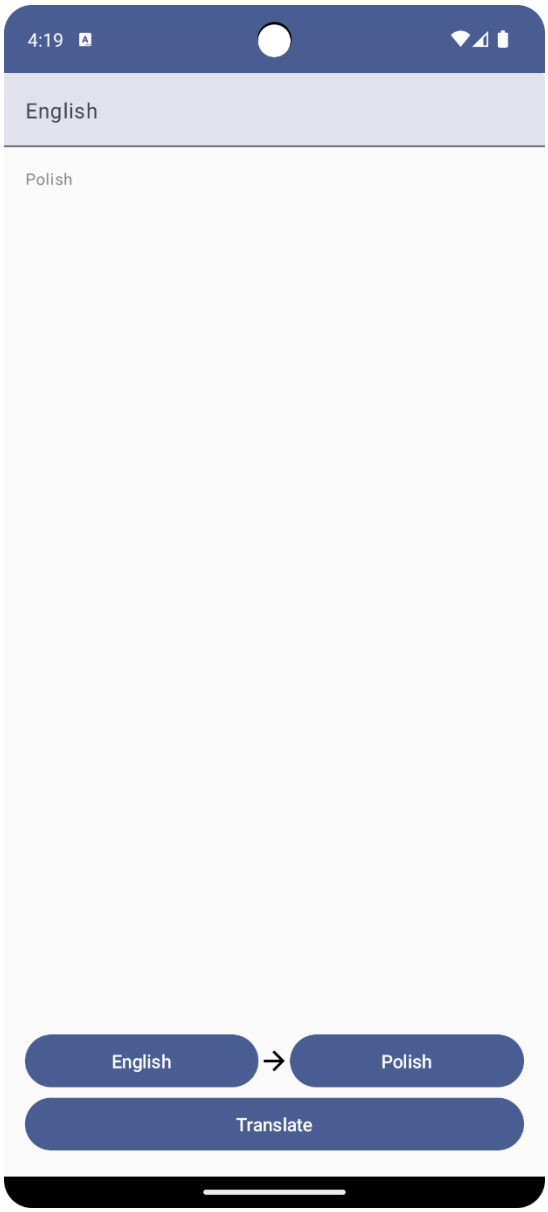
Kolejnym krokiem jest zainicjowanie pobierania modelu. Model zostanie pobrany tylko w przypadku, gdy nie został znaleziony na urządzeniu. Gdy model jest dostępny aplikacja przechodzi do rzeczywistego tłumaczenia tesktu w oparciu o pobrany model. Zakończenie procesu zakończy się wyświetleniem stosownego komunikatu typu ***Toast*** zarówno w przypadku sukcesu jak i wystąpienia błędu.

```
    translator.downloadModelIfNeeded(downloadConditions)
        .addOnSuccessListener { it: Void!
            Log.d(TAG, msg: "startTranslation: model ready, start translation...")

            dialogMessage("Translating...")

            translator.translate(inputData)
                .addOnSuccessListener{translatedText ->
                    Log.d(TAG, msg: "startTranslation: translatedText: $translatedText")

                    showDialog(false)
                    translatedText(translatedText)
                }
                .addOnFailureListener{e->
                    showDialog(false)
                    Log.e(TAG, msg: "startTranslation: ", e)

                    showToast("Failed to translate due to ${e.message}")
                }
        }
        .addOnFailureListener {e->
            showDialog(false)
            Log.e(TAG, msg: "startTranslation: ", e)

            showToast("Failed due to ${e.message}")
        }

}
```

Rys.4 - Pobieranie modelu

3. Prezentacja działania aplikacji

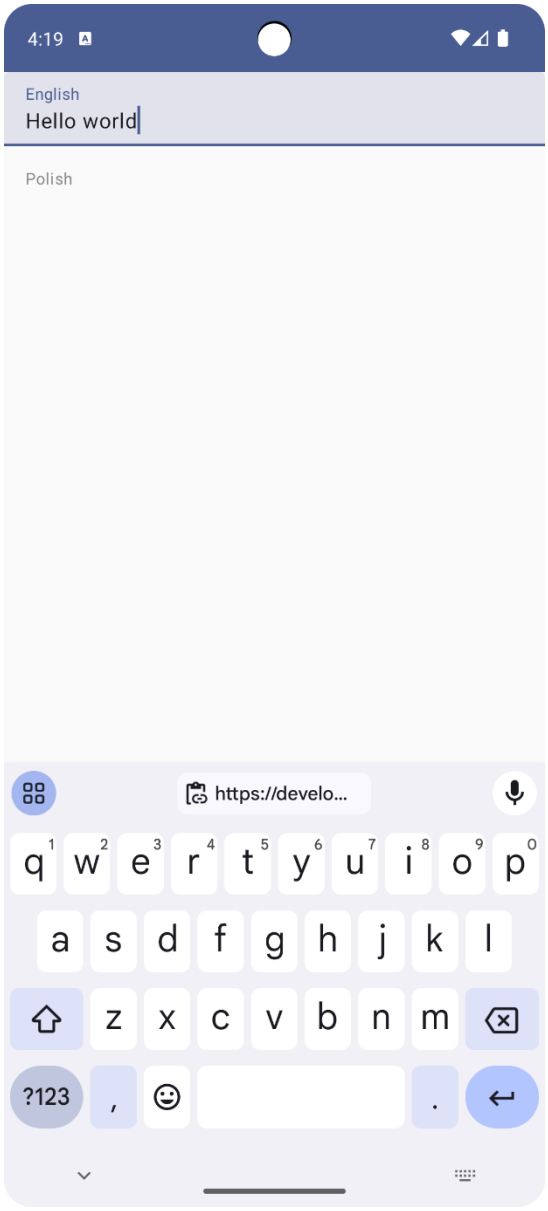
Widok GUI dla aplikacji translatora:



Rys.6 - Interfejs użytkownika aplikacji

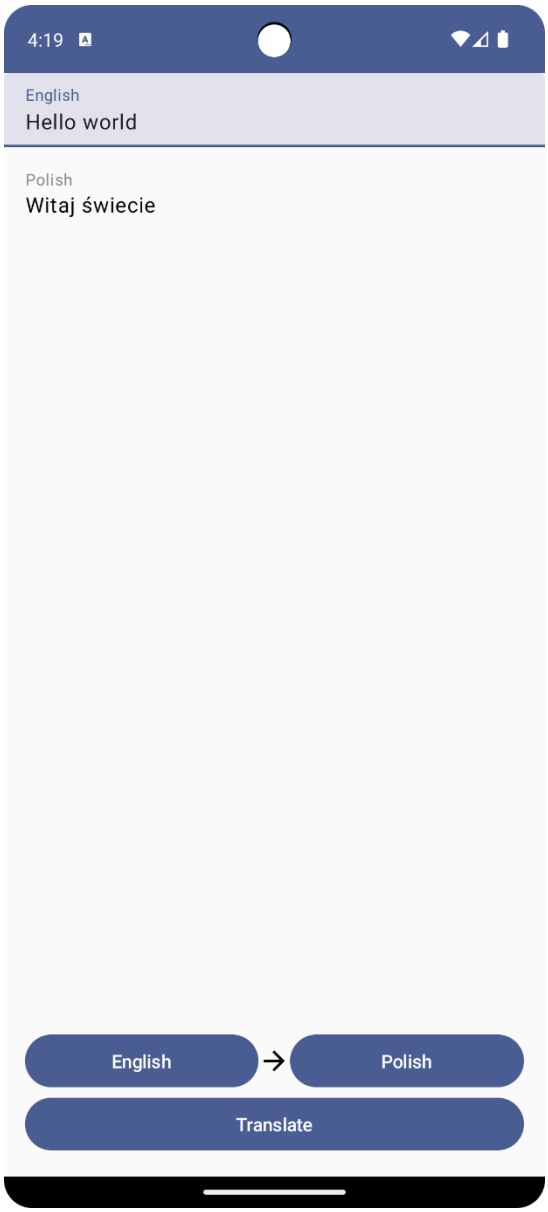
Scenariusz I: Tłumaczenie z języka angielskiego na język polski.

W górnym polu tekstowym należy wpisać tekst, który ma zostać przetłumaczony na wybrany język.



Rys.7 - Użycie aplikacji

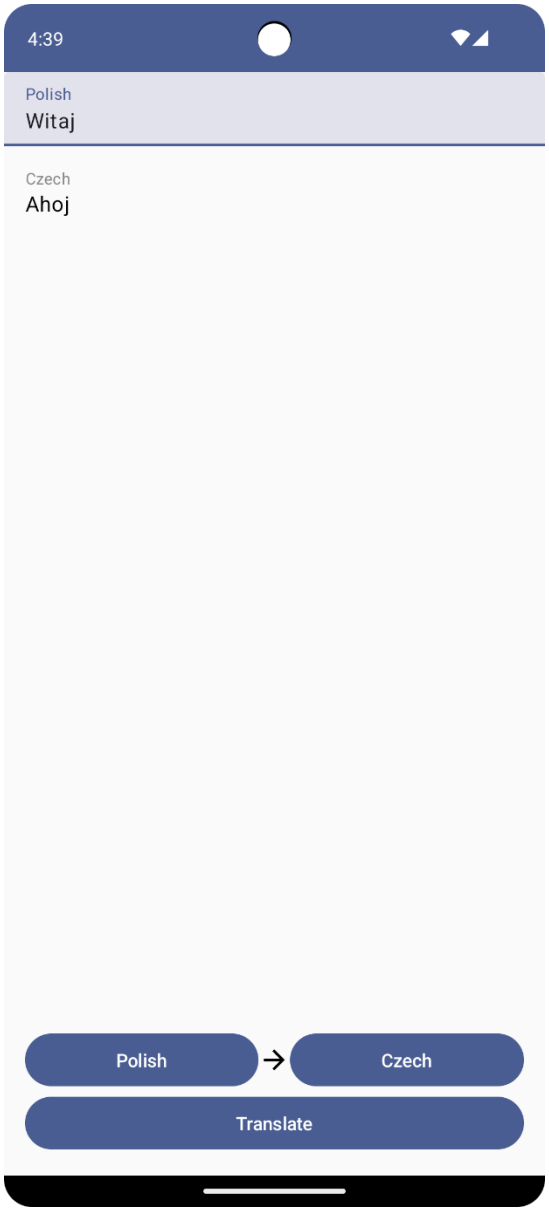
W dolnym polu tekstowym znajduje się wynik tłumaczenia. Podczas opisywanego testu przetłumaczono z języka angielskiego “Hello world” na język polski, co dało wynik “Witaj świecie”.



Rys.8 - Tłumaczenie na język polski

Scenariusz II: Tłumaczenie z języka polskiego na język czeski.

W celu przetłumaczeniu na dany język, na telefon z systemem android musi zostać pobrany wymagany pakiet językowy. Podczas opisywanego testu przetłumaczono z języka polskiego “Witaj” na język czeski, co dało wynik “Ahoj”.



Rys.9 - Tłumaczenie na język czeski

4. Podsumowanie i wnioski końcowe

Zastosowanie Kotlina w tworzeniu aplikacji symulującej translator pozwala na uzyskanie wydajnego, bezpiecznego i łatwego w utrzymaniu rozwiązania. Dzięki zaawansowanym technikom przetwarzania języka naturalnego oraz integracji z zewnętrznymi usługami tłumaczeniowymi, aplikacja może zapewnić użytkownikom wysokiej jakości tłumaczenia w czasie rzeczywistym.

5. Źródła

Biblioteka ML Kit

https://developers.google.com/ml-kit/language/translation?hl=pl
https://developers.google.com/ml-kit/language/translation/android?hl=pl