Narzędzie do rozpoznawania języka oparte na bibliotece fastText

Autorzy:

Tomasz Macałka

Michał Pieczara

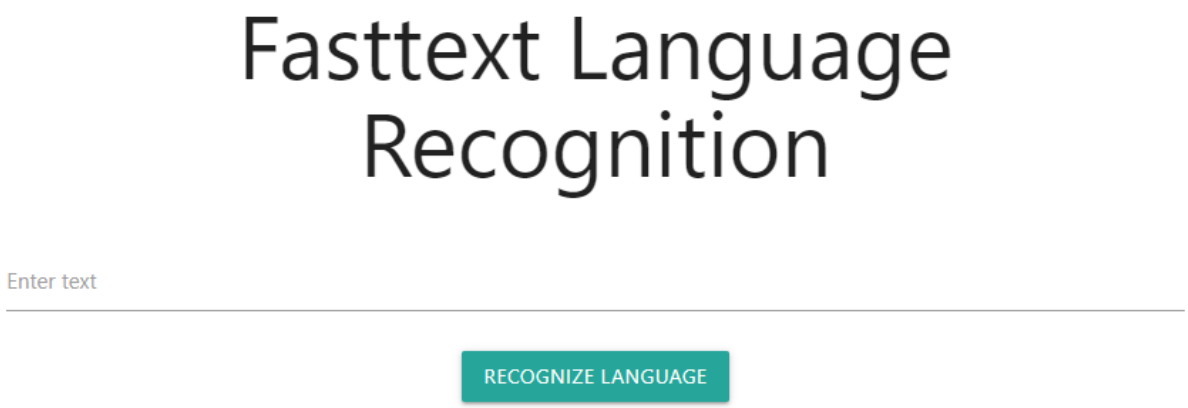
Opracowywany projekt, dotyczy przygotowania programu umożliwiającego automatyczne rozpoznawanie języka we wprowadzonym tekście. Użyty w celach zbudowania aplikacji gotowy model fasttext[1], rozpoznający 176 języków został wykorzystany wraz z modelem wytrenowanym przez autorów projektu, który rozpoznaje trzy języki w tekście - angielski, polski i niemiecki. Modele te użyte równolegle pozwalają na porównanie precyzji i czułości klasyfikowania tekstu do danej etykiety języka.

Celem projektu jest opracowanie skutecznego i efektywnego rozwiązania umożliwiającego automatyczne rozpoznawanie języka w tekście. Problem ten ma duże znaczenie w różnych dziedzinach, takich jak tłumaczenie, analiza sentymentu czy personalizacja treści. Dzięki automatycznemu rozpoznawaniu języka możliwe jest szybkie i precyzyjne przetwarzanie tekstu w zależności od jego języka, co przyczynia się do usprawnienia wielu procesów komunikacyjnych i analizy tekstowej.

W ramach tego programu, zespół przygotował kompleksowe rozwiązanie umożliwiające rozpoznawanie języka w tekście. Przedstawimy teraz najważniejsze elementy tego rozwiązania.

W naszym projekcie opracowaliśmy aplikację, która umożliwia interakcję z użytkownikiem poprzez interfejs graficzny (rys. 1). Użytkownik ma możliwość wprowadzenia dowolnego tekstu, a aplikacja automatycznie rozpoznaje język tego tekstu i prezentuje wyniki. Interfejs ten został opracowany przy użyciu Flask[2]- mikro frameworka aplikacji webowych stworzonego w języku Python. W ramach części stworzonej w Flask przy przejściu na stronę główną jest renderowany szablon strony index.html. Następnie, gdy użytkownik wprowadzi dane w formularzu i wystąpi metoda “POST” wprowadzony tekst zostaje tłumaczony z wykorzystaniem załadowanych modeli. W widoku zostaje zwrócony wynik tłumaczenia dla dwóch najbardziej prawdopodobnych języków oraz informacje na temat każdego użytego modelu (ilość rekordów do przetestowania modelu, precyzja i czułość).

[1] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of Tricks for Efficient Text Classification
[2] Flask Python Module Documentation. Dostępne online: https://flask.palletsprojects.com/en/2.3.x/ (Ostatni dostęp: 14.05.2023)

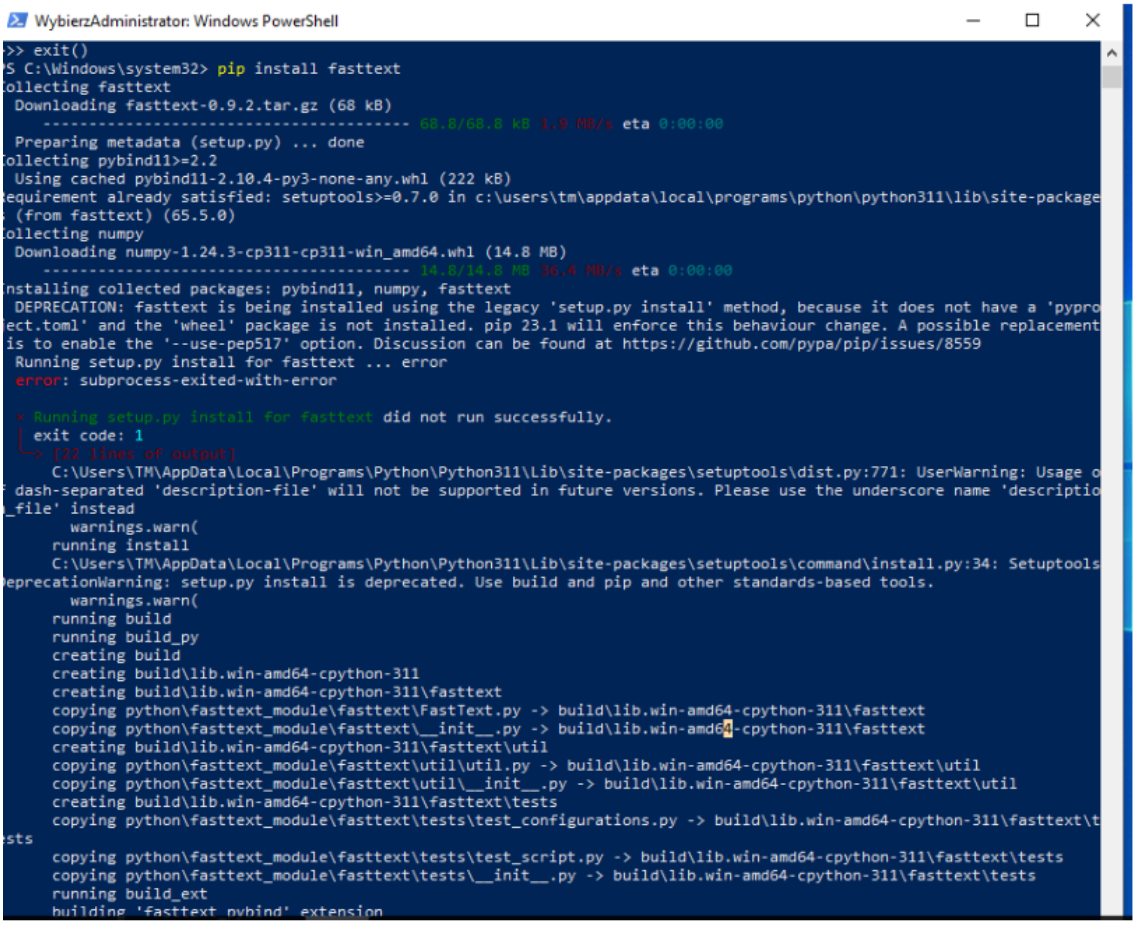


Rys.1. Wygląd interfejsu graficznego.

Część odpowiedzialna za tłumaczenie powstała przy użyciu biblioteki fastText[3]. W celu jej wykorzystania należy zainstalować moduł przy użyciu polecenia “pip install fasttext”. Korzystając z systemu operacyjnego Windows można spotkać się z błędami instalacji (rys. 2), które wynikają z brakujących bibliotek “Visual Build Tools”. W przypadku naszego zespołu problem występował nawet po ich zainstalowaniu. W związku z tym środowisko został skonfigurowane przy użyciu dystrybucji Linux - środowiska skonteneryzowanego python:3.11-slim. Dodatkowo w pliku requirements.txt zostały zdefiniowane wersje użytych bibliotek fastText 0.9.2 i Flask 2.3.2 (lub nowsze kompatybilne). Dzięki temu bez problemu można zainstalować i odtworzyć aplikację na dowolnym urządzeniu.

Po skonfigurowaniu środowiska część programu odpowiedzialna za tłumaczenie została podzielona na kilka etapów. Pierwszym etapem było przygotowanie plików zawierających zdania do wytrenowania własnego modelu. Pobrane zdania pochodziły z portalu tatoeba.org. W tym celu należało odpowiednio przygotować pliki, tak, aby były one zgodne z oczekiwanym przez fastText formatem. Format z portalu tatoeba (rys. 3) został dostosowany do zgodnego formatu (rys. 4) przy użyciu funkcji re.sub (rys. 5), która podmienia fragmenty tekstu przy dopasowaniu za pomocą wyrażeń regularnych.

[3] fastText Python Module Documentation. Dostępne online: https://fasttext.cc/docs/en/python-module.html (Ostatni dostęp: 14.05.2023)



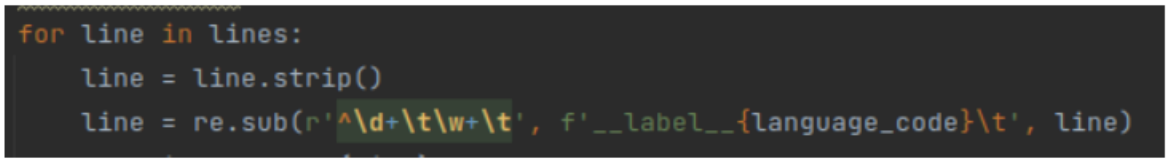
Rys. 2. Błędy podczas instalacji biblioteki fastText z poziomu Windows.

```
1276    eng Let's try something.
1277    eng I have to go to sleep.
1280    eng Today is June 18th and it is Muiriel's birthday!
1282    eng Muiriel is 20 now.
1283    eng The password is "Muiriel".
1284    eng I will be back soon.
1286    eng I'm at a loss for words.
```

Rys. 3. Zdania pobrane z portalu tatoeba.org i ich format.

```
__label__en Let's try something.
__label__en I have to go to sleep.
__label__en Today is June 18th and it is Muiriel's birthday!
__label__en Muiriel is 20 now.
__label__en The password is "Muiriel".
__label__en I will be back soon.
__label__en I'm at a loss for words.
```

Rys. 4. Zdania dostosowane do formatu oczekiwanego przez fastText.



Rys. 5. Kod odpowiedzialny za dostosowanie formatu danych.

Po tym zabiegu zgodnie z konfiguracją aplikacji pierwsze 50000 rekordów z każdego z trzech języków zostało zapisane do wspólnego pliku combined.txt jako zbiór zdań do trenowania modelu. Ostatnie 20000 rekordów z każdego z trzech języków zostało natomiast zapisane do wspólnego pliku test_sentences.txt jako zbiór zdań do testowania modelu.

Dzięki przygotowaniu plików w ten sposób proces trenowania modelu uruchamia się za pomocą polecenia fasttext.train_supervised wraz z odpowiednimi parametrami, których wartości były wielokrotnie zmieniane, aby uzyskać jak najbardziej zadowalające wyniki (rys. 6). Funkcja train_supervised przyjmuje następujące parametry:

- input - ścieżka do pliku z danymi treningowymi,
- label_prefix - prefiks używany przed etykietami klas w pliku z danymi,
- epoch - liczba epok treningowych - wartość 25 - zbyt mała wartość - doprowadza do niedouczenia modelu, natomiast zbyt duża wartość może prowadzić do przeuczenia modelu, czyli sytuacji, w której model nauczy się zbytnio dostosowywać do danych treningowych i będzie słabo generalizował na nowych danych, co może prowadzić do niedokładnych predykcji na danych testowych lub w praktyce,
- lr - learning rate - wartość 0.1 - zbyt mała wartość może spowodować, że model będzie uczył się zbyt wolno, co wydłuży czas treningu, natomiast zbyt duża wartość może spowodować, że model będzie miał problem z generalizacją i osiągnie gorsze wyniki na danych testowych,
- wordNgrams - długość n-gramów używanych do reprezentacji słów - wartość 1 - zbyt mała wartość może prowadzić do obniżenia jakości detekcji języka, natomiast zbyt duża wartość może prowadzić do zwiększenia wymiarowości wektorów słów, co zwiększa ilość parametrów do nauki i może prowadzić do przetrenowania modelu,
- bucket - liczba kubełków używanych do haszowania - wartość 2000000 - zbyt mała wartość może prowadzić do kolizji haszy, co oznacza, że dwa różne słowa zostaną przypisane do tego samego kubełka i będą traktowane jako jedno słowo, to może prowadzić do pogorszenia jakości modelu i wynikającej z niego analizy. Zbyt duża

wartość może prowadzić do niepotrzebnego zwiększenia czasu trenowania i pamięci potrzebnej do przechowywania modelu,

- dim - liczba wymiarów reprezentacji wektorowej - wartość 300 - zbyt mała wartość może spowodować, że model nie będzie w stanie zachować odpowiedniej ilości informacji o słowach, co wpłynie na jakość klasyfikacji. Zbyt duża wartość może prowadzić do overfittingu, tj. model będzie zbyt dobrze dopasowany do zbioru treningowego, a nie będzie w stanie dobrze generalizować na nowych danych,
- thread - liczba wątków używanych do treningu - wartość 4.

Rys. 6. Funkcja użyta do trenowania modelu i postęp wyświetlający się w konsoli.

Po procesie nauki własnego modelu został załadowany gotowy model lid.176.ftz pobrany ze strony internetowej fasttext.cc. Ładowanie odbywa się przy użyciu funkcji fasttext.load_model. W następnej kolejności zostały wykonane testy precyzji i czułości modeli wywołując instancję danego modelu z metodą test, np. own_model.test('static/test_sentences.txt').

Gdy modele są już załadowane i przetestowane, a dane użytkownika z formularza są odebrane można przejść do procesu rozpoznawania języka. W tym celu wywołuje się instancję danego modelu z metodą predict, np. own_model.predict(text, k=2), gdzie text to tekst użytkownika, a k to liczba rozpoznanych etykiet wraz z ich prawdopodobieństwem.

Dane otrzymane z funkcji testujących i rozpoznających język są wybierane ze zmiennych poprzez odwołanie się do rekordów o odpowiadających im indeksach i zwracane do funkcji renderującej szablon index.html

W rezultacie, opracowane narzędzie do rozpoznawania języka oparte na bibliotece fastText stanowi skuteczne rozwiązanie, które może być wykorzystane w różnych aplikacjach wymagających analizy języka naturalnego. Otrzymane wyniki testów precyzji wyniosły

około 98% co stanowi bardzo dobry wynik, jak na tak niewielki zbiór danych użyty do treningu własnego modelu. W niektórych przypadkach wytrenowany przez nasz zespół model był nawet bardziej precyzyjny od gotowego modelu dostarczanego przez fasttext. Użyta przez nas biblioteka jest potężnym narzędziem i oprócz rozpoznawania języka można śmiało ją wykorzystywać do wielu innych dziedzin przetwarzania języka naturalnego, takich jak analiza sentymentu, klasyfikacja tekstu, tłumaczenie czy personalizacja treści.

Stworzona aplikacja dostępna jest pod adresem:
http://macalka.duckdns.org:5000
Repozytorium kodu dostępne jest pod adresem:
https://github.com/teh42eem00/PJN_Language_Recognition

Bibliografia

1. A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, *Bag of Tricks for Efficient Text Classification*
2. *fastText Python Module Documentation,* online: https://fasttext.cc/docs/en/python-module.html (Ostatni dostęp: 14.05.2023)
3. *Flask Python Module Documentation,* online: https://flask.palletsprojects.com/en/2.3.x/ (Ostatni dostęp: 14.05.2023)