14.05.2023, Kraków

Narzędzie do rozpoznawania języka w oparciu o bibliotekę fasttext

Przygotowali:
Jan Wielgosiński
Piotr Zych

Cel projektu

Celem projektu było przygotowanie narzędzia, które będzie rozpoznawać język, tekst w oparciu o bibliotekę fasttext. Klasyfikacja tekstu jest kluczowym problemem dla wielu aplikacji w aktualnym czasie, takich jak wykrywanie spamu, analiza sentymentu czy wykrywanie języka.

Czym jest biblioteka fasttext?

Biblioteka fasttext jest biblioteką typu open-source stworzoną przez zespół Facebook AI Research (FAIR), która pozwala na klasyfikację tekstu. W swojej początkowej wersji biblioteka ta wykorzystywała technikę trenowania modelu o nazwie skip-gram, ale aktualnie wspiera również metodę Continuous Bag of Words (CBOW). Fasttext pozwala na trenowanie około 1 miliarda słów w czasie mniejszym niż 10 minut.

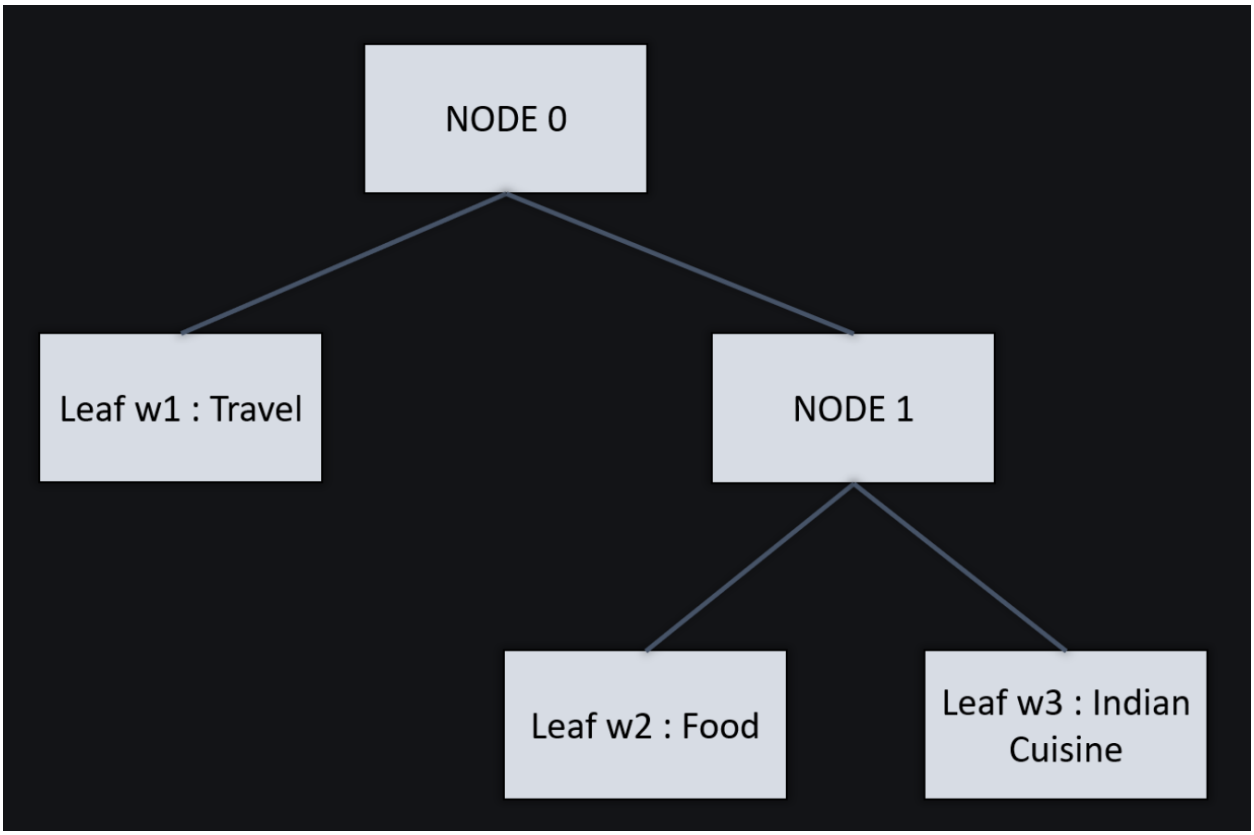
Klasyfikacja tekstu

Celem klasyfikacji tekstu, jest przypisanie dokumentów do jednej lub wielu kategorii. Takimi kategoriami mogą być język danego tekstu, oceny recenzji czy również jedzenie. Obecnie jednym z dominujących podejść do budowy tego typu klasyfikatorów jest wykorzystywanie do tego uczenia maszynowego, czyli przekazanie sztucznej inteligencji reguł klasyfikacji na podstawie przykładów. Do budowy takich klasyfikatorów potrzebne są dane pogrupowane, które składają się z dokumentów oraz odpowiadających im kategorii czy tagów bądź etykiet.

Rodzaje klasyfikatorów

Klasyfikator liniowy polega na tym, że tekst oraz etykiety są reprezentowane jako wektory. Znajdujemy takie reprezentacje wektorowe, których tekst i związane z nim etykiety mają zbliżone wektory. Tłumacząc dokładniej to wektor, który odpowiada danemu tekstowi jest bliższy odpowiadającej jej etykiecie.

Klasyfikator hierarchiczny polega na tym, że reprezentowane etykiety są przedstawione w formie drzewa binarnego. Każda gałąź w drzewie binarnym reprezentuje prawdopodobieństwo. Etykieta natomiast jest reprezentowana przez prawdopodobieństwo wzdłuż ścieżki do danej etykiety. Oznacza to, że węzły liści reprezentując etykiety. Biblioteka fasttext używa w tym celu algorytmu Huffmana do budowy drzew, aby w pełni wykorzystać możliwość, że klasy mogą być niezbalansowane, niewyważone.



Rysunek 1: Drzewo binarne reprezentowanych etykiet
Źródło: https://www.geeksforgeeks.org/fasttext-working-and-implementation/

Konfiguracja

Jest kilka sposobów konfiguracji biblioteki fasttext. W tym projekcie został zaimplementowany poprzez zainstalowanie biblioteki poprzez kompilator kodu za pomocą polecenia:

Rysunek 2: Import biblioteki fasttext do pliku pythonowego

Pobranie biblioteki w taki sposób pozwoli na wykorzystanie jej możliwości do rozpoznawania danej kategorii, którą oczekujemy w naszym programie.

Kolejnym sposobem na zainstalowanie oraz korzystanie z pełni możliwości biblioteki fasttext jest wykonanie poniższych poleceń:

```
git clone https://github.com/facebookresearch/fastText.git
cd fastText
make
```

Przygotowanie danych

W przypadku przygotowania danych, aby biblioteka fasttext była w stanie się ich nauczyć, potrzebne jest je przekazanie w odpowiednim formacie. Biblioteka ta oczekuje, aby kategoria była w pierwszej kolumnie z prefiksem „__label__” przed nazwą kategorii. Jeśli chcielibyśmy utworzyć kategorię z językiem polskim, musiała ona by wyglądać w taki oto sposób: __label_pl To jest przykładowy tekst. Oczywiście przy przygotowywaniu danych jest możliwe, aby dany tekst należał do kilku kategorii, na przykład: __label__en __label__chocolate American equivalent for British chocolate terms. Takim sposobem możemy zauważyć, że taki tekst będzie przypisany do dwóch kategorii, do kategorii język angielski oraz kategoria czekolada.

Sposób działania

W pierwszym przykładzie działania zaprezentowane jest w jaki sposób biblioteka fasttext wykorzystuje do rozpoznawania danego języka modelu zaimplementowanego w formie pliku o nazwie lid.176.ftz.

```
1  from pprint import pprint
2
3  import fasttext
4
5  model_file = "lid.176.ftz"
6  model = fasttext.load_model(model_file)
7  text = ["To jest przykładowy tekst", "Yo soy polaco"]
8  for t in text:
9      predicted_language = model.predict(t)
10     pprint(predicted_language)
11
```

Rysunek 3: Fragment kodu narzędzia do rozpoznawania języka

Poniżej zaprezentowany jest wynik działania powyższego kodu. Możemy zauważyć, że w zaimplementowanym kodzie znalazły się dwa zdania, jedno w języku polskim, a drugie w języku hiszpańskim. Wynik, który nam przekazał uruchomiony kod jest predykcją rozpoznania tekstu przy użyciu modelu. Jak widać rozpoznał oba zdania w 99% zdanie w języku polskim, a w 98% zdanie w języku hiszpańskim.

```
(('__label__pl',), array([0.99930513]))
(('__label__es',), array([0.98353422]))

Process finished with exit code 0
```

Rysunek 4: Wynik działania narzędzia

Dane trenowane oraz testowe

Zanim zaczniemy trenować model, dobrym rozwiązaniem będzie podzielenie danych, które będą danymi do trenowania oraz testowania. W celu przeprowadzenia eksperymentu został wykorzystany plik o nazwie cooking.stackexchange.txt. Dzięki poleceniu wc -l cooking.stackexchange.train jesteśmy w stanie dowiedzieć się, że ten plik zawiera 15404 linii tekstu. Możemy wykorzystać 80% danych do trenowania oraz 20% do testowania, czyli w przybliżeniu linii do wytrenowania naszego modelu będzie wynosić 12324 linie, a pozostałe będą w formie testowych. Aby wydzielić nasze dane do eksperymentu musimy użyć dwóch poleceń:

```
head -n 12324 cooking.stackexchange.train > training.txt
tail -n 3080 cooking.stackexchange.train > testing.txt
```

Kolejnym krokiem będzie nauczenie naszego modelu tych słów, których chcemy, aby został wytrenowany:

```
./fasttext supervised -input training.txt -output
cooking_question_classification_model
```

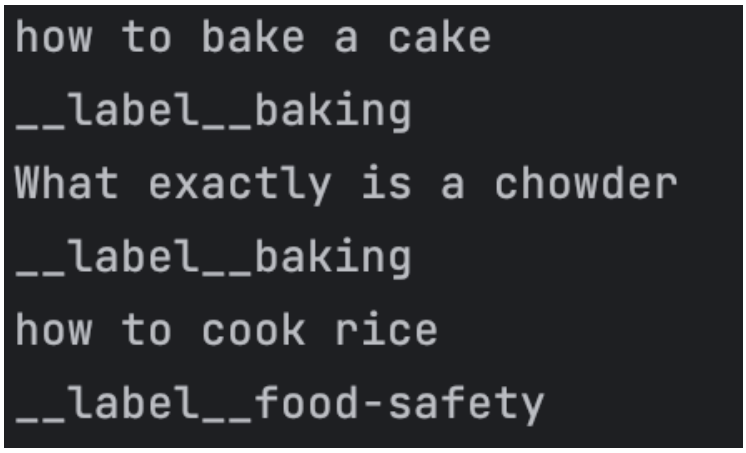
```
Read 0M words
Number of words:  14492
Number of labels: 735
Progress: 100.0% words/sec/thread:   37127 lr:  0.000000 avg.loss: 10.225876 ETA:   0h 0m 0s
```

Rysunek 5: Wynik uruchomienia powyższego polecenia

Model został wytrenowany dzięki temu i może teraz zweryfikować czy będzie odpowiednio klasyfikował nasze zapytania. Aby uruchomić możliwość zadawania pytań modelowi musimy wykonać poniższe polecenie:

```
./fasttext predict cooking_question_classification_model.bin –
```

Znak pauzy na końcu (-) pozwoli nam wypisywać dane w konsoli, które potem model będzie weryfikował. Poniżej zamieszczone są przykładowe wyniki wytrenowanego modelu:



Rysunek 6: Wynik wypisania zapytań oraz predykcji modelu

Podsumowanie

Patrząc w przyszłość wiemy, że wektory słów są niesamowicie potężną koncepcją i technologią, która umożliwi znaczący przełom w zastosowaniach i badaniach PJN. Podkreślają one moc wyuczonych reprezentacji danych wejściowych w warstwach ukrytych. Budowanie lepszych aplikacji PJN nieuchronnie wymaga dobrego zrozumienia wektorów słów, a biblioteka fasttext w znacznym stopniu pomoże w rozwoju tego zrozumienia.

Bibliografia

https://fasttext.cc
https://www.geeksforgeeks.org/fasttext-working-and-implementation/
https://pythonwife.com/fasttext-in-nlp/