Zastosowanie analizy SHAP w analizie sentymentu metodami NLP

Autorzy:
Bartosz Konopka, Szymon Kolanko i Michał Partyka

Link do projektu:
https://github.com/BKonopka123/MIO_project_nlp_shap

1. Opis projektu

Celem projektu jest zastosowanie analizy SHAP (SHapley Additive exPlanations) do wyjaśnienia wyników modelu analizy sentymentu w NLP (Natural Language Processing). Analiza SHAP pozwala na zrozumienie, jak poszczególne cechy wpływają na decyzje modelu, co jest szczególnie ważne w kontekście modeli złożonych, takich jak te używane w NLP.

2. Cele projektu

- Zastosowanie technik NLP do analizy sentymentu.
- Zrozumienie i wyjaśnienie predykcji modelu analizy sentymentu za pomocą analizy SHAP.
- Ocena, w jaki sposób poszczególne cechy (np. słowa, frazy) wpływają na ostateczny wynik analizy sentymentu.

3. Dokumentacja kodu

3.1 Wczytanie i preprocessing

Preprocessing danych jest kluczowym etapem, który wpływa na efektywność i dokładność modelu. Poprawnie przeprowadzony

preprocessing może znacznie poprawić wyniki modelu i zapewnić jego stabilność oraz zdolność generalizacji. W ramach preprocessingu wykonaliśmy:

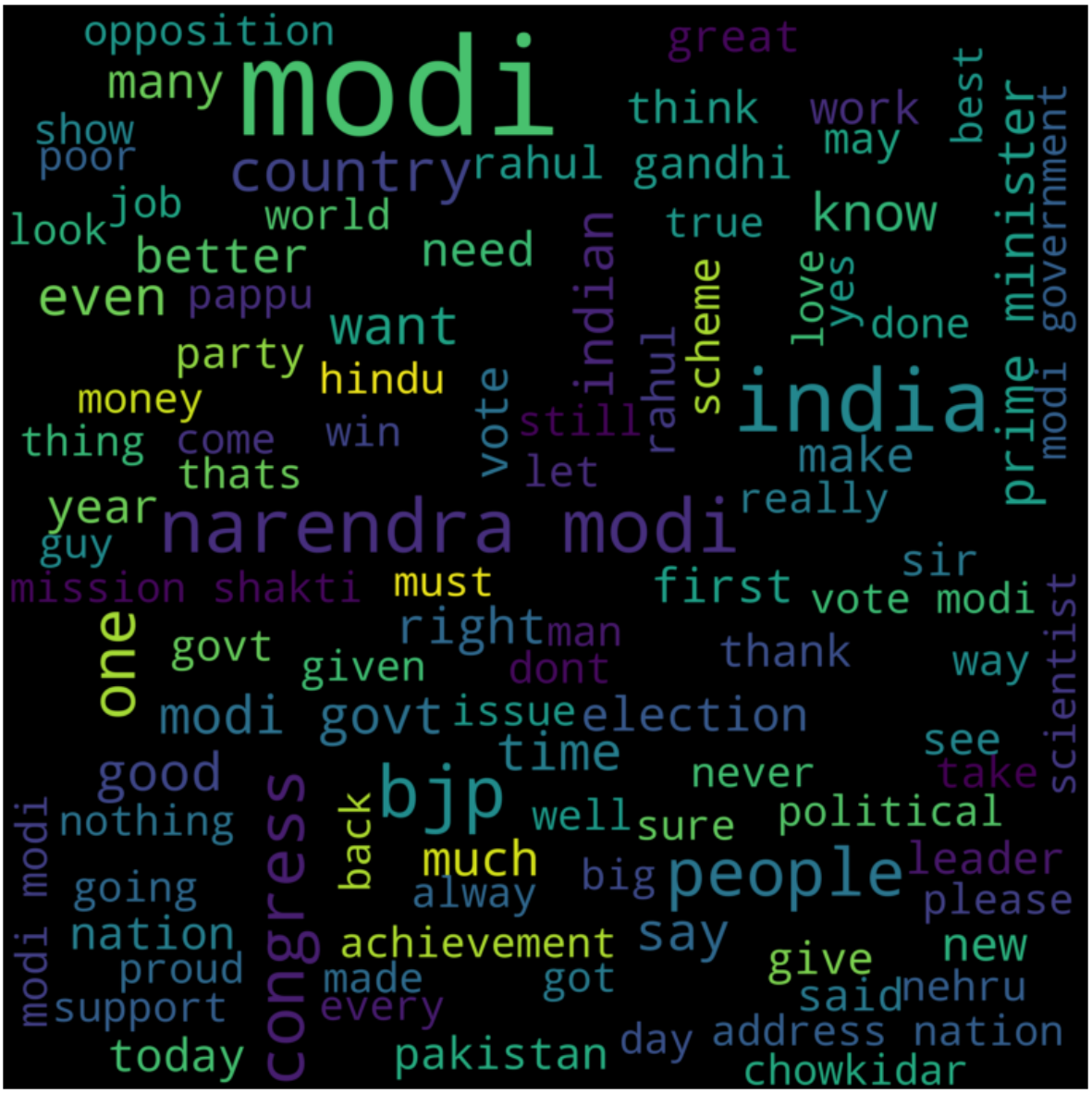
- Wczytanie danych z pliku *Twitter.csv*.
- Usuwanie URL'ów z treści twittów.
- Usuwanie wzmianek (@).
- Usuwanie znaków interpunkcyjnych.
- Usuwanie emotikonów.
- Usuwanie powtarzających sie znaków.
- Usuwanie stop words w języku angielskim.
- Ustawienie wszystkich liter na małe.

3.2. Prezentacja wstępnych danych

- Przedstawienie przykładowych danych.

	text	sentiment
7296	raga said india enough money scheme wonder party couldn' launch upa government didn' even enough money orop buy rafales mean india five years modi govt enough money	negative
39675	2002 gujarat genocide vajpayee wanted modi stripped primary membership advani pleaded case got retained vajpayee modi failed rajdharma	negative
12224	republic channel sharing news leading indian representing india factorage tyrant modi every time sharing modis heavily funded modis sarkar gewgaw despot modis regime	negative
20859	rishi bhai lost 100 followers want knw modi fans punished twitter	neutral
12700	ppl started getting modis 600 instalment whereas congress promises never got fulfilled since 1947 till	neutral

- Przedstawienie najczęściej występujących słów.



Przykład dla tweet'ów pozytywnych.

3.3. Trenowanie modelu

3.3.1 Oryginalny model Pytorch

Pierwszy model został stworzony przy użyciu biblioteki PyTorch. Model został zdefiniowany jako obiekt dziedzicząca po klasie Module, składa się z warstwy osadzeń (nn.Embedding), która przekształca tokeny wejściowe na wektory osadzeń, sieci LSTM (nn.LSTM), która przetwarza sekwencje osadzeń, warstwy dropout (nn.Dropout), która pomaga zapobiegać przeuczeniu, warstwy w pełni połączonej (nn.Linear), która przekształca wyjście LSTM na wyniki dla każdej klasy, i funkcji softmax (nn.Softmax), która przekształca wyniki na prawdopodobieństwa przynależności do klasy. Do tokenizacji zdań wejściowych został użyty tokenizer z modułu Keras a ich długość została wyrównana do tej samej długości. Model ten został przez nas porzucony ze względu na niemożność przeprowadzenia analizy SHAP. Napotkany błąd okazał się niemożliwy do obejścia i wynikał z bug'u implementacji modułu SHAP. Z tego powodu przenieśliśmy się na bibliotekę Keras, która znacznie lepiej współpracuje z analizą SHAP.

3.3.2 Drugi model keras.Sequential

Model jest inicjalizowany jako Sequential() z biblioteki keras, ponieważ niestety z powodu braku czasu, oraz problemów związanych z analizą SHAP, nie zdołaliśmy poprawnie dokończyć własnego, pierwotnego modelu. W poniższych punktach wytłumaczone jest również podejście dla oryginalnego modelu.

- Inicjalizacja modelu:

  Inicjalizujemy model sekwencyjny, dodajemy do niego warstwy takie jak Embedding, SpatialDropout1D, LSTM i Dense. Konfigurujemy każdą warstwę odpowiednio do naszych potrzeb. W oryginalnym modelu również dodajemy warstwy, jednakże nie pokrywają się one dokładnie z tymi użytymi dla modelu sekwencyjnego.

- Zakodowanie etykiet tekstowych na numeryczne:

  Używamy *Tokenizer* do przekształcenia tekstu w sekwencje numeryczne, a następnie te sekwencje są dopasowywane do tej samej długości za pomocą *pad_sequences*. Etykiety sentymentu są zakodowane na numeryczne wartości. W oryginalnym modelu wykorzystujemy *Tokenizer* w podobny sposób.

- Podział danych na zbiory testowe i treningowe:

  Dane są podzielone na zbiory treningowe i testowe w stosunku 80:20.

- Przygotowanie maski uwagi:

  W ostatecznej implementacji modelu maska uwagi nie są przygotowywane, aczkolwiek w oryginalnym modelu używamy funkcji *np.where*, aby stworzyć maskę, która ma wartość 1 w miejscach, gdzie tokeny nie są zerami (są rzeczywiste), i 0 w miejscach paddingu (są zerami). Maska nastepnie jest konwertowana na tensor.

- Konwersja danych na tensory:

Dane wejściowe i etykiety są konwertowane na tablice NumPy (odpowiednik tensorów w oryginalnym modelu) przed przekazaniem ich do modelu.

- Wyważanie danych:

  W modelu sekwencyjnym wyważanie danych nie jest wymagane, jednakże w oryginalnym modelu na podstawie krotności występowania każdej z klas przypisujemy im różne wagi, tak aby zrównoważyć dane.

- Ustawienie optymalizatora na adam:

  W obu podejściach wykorzystujemy optymalizator adam.

- Trening:

  Model jest trenowany na zbiorze treningowym przez określoną liczbę epok. EarlyStopping jest używany do zatrzymania treningu, jeśli wydajność modelu na zbiorze walidacyjnym przestanie się poprawiać przez 3 kolejne epoki.

Otrzymane wyniki:

Model stale poprawia się w nauce danych treningowych. Od około piątej epoki model zaczyna wykazywać stabilizację wyników na zbiorze walidacyjnym, a w niektórych epokach nawet pogorszenie wyników, co może sugerować przeuczenie. Strata na zbiorze walidacyjnym oscyluje wokół wartości 0.55, co może wskazywać na pewne trudności modelu w dalszej poprawie generalizacji. Model wykazuje obiecujące wyniki, jednakże dalsze optymalizacje są potrzebne, aby poprawić jego generalizację i stabilność na zbiorze walidacyjnym. Również z powodu braku czasu, nie udało nam się ostatecznie tych zmian wprowadzić.

4. Analiza SHAP

4.1. Wczytanie i przygotowanie danych do analizy SHAP

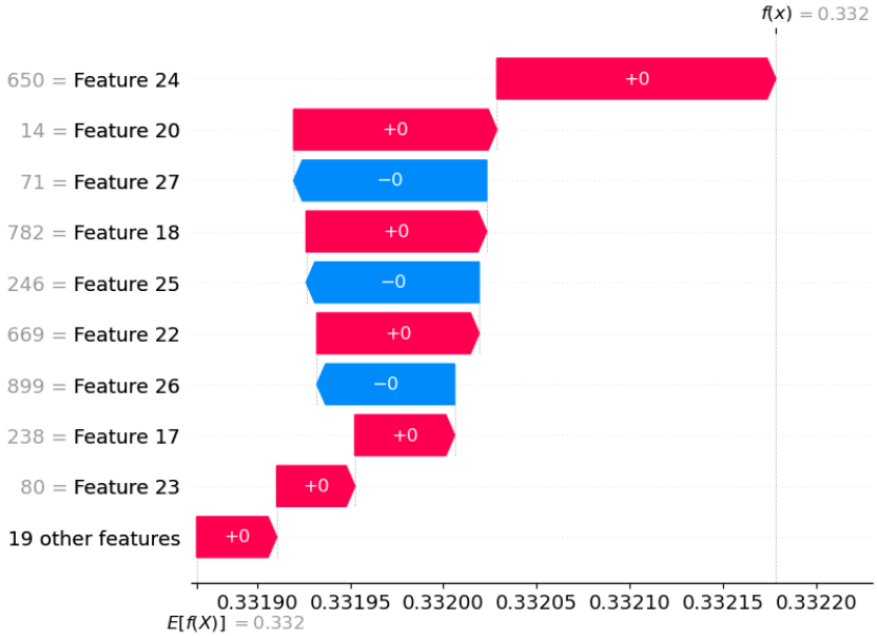
- Wczytujemy model, aby móc na nim przeprowadzić dalsze analizy.
- Przygotowujemy tokenizer do przetwarzania tekstu.
- Wczytujemy plik *realdonaldtrump.csv* z tweet'ami, a następnie czyścimy potrzebne nam dane.
- Przekształcamy tekst na sekwencje numeryczne do modelu.

4.2. Przygotowanie tła i obliczanie wartości SHAP

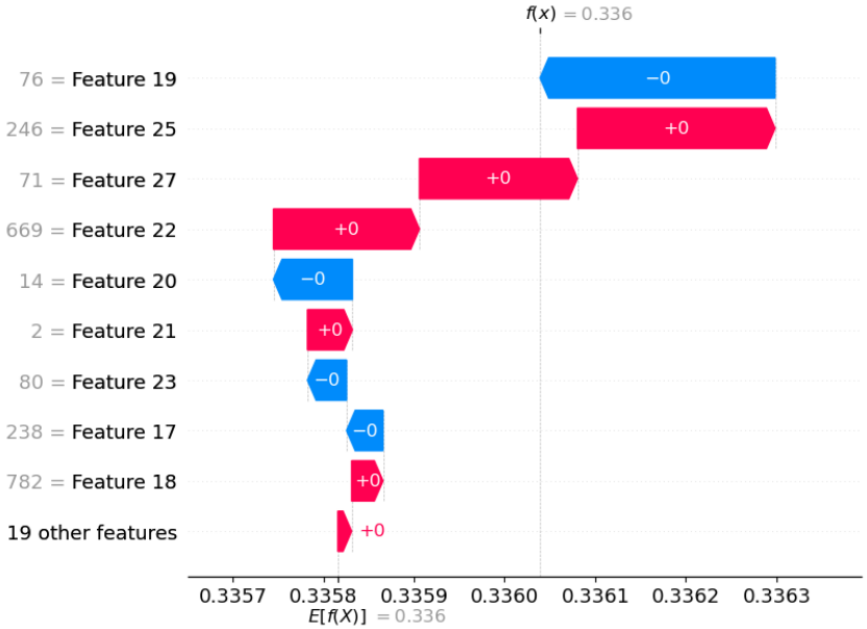
- Wybieramy reprezentatywny zestaw danych jako tło.
- Inicjalizujemy SHAP Explainer do analizy modelu.
- Obliczamy wartości SHAP, aby zrozumieć wpływ poszczególnych cech na przewidywania modelu.

5. Wyniki

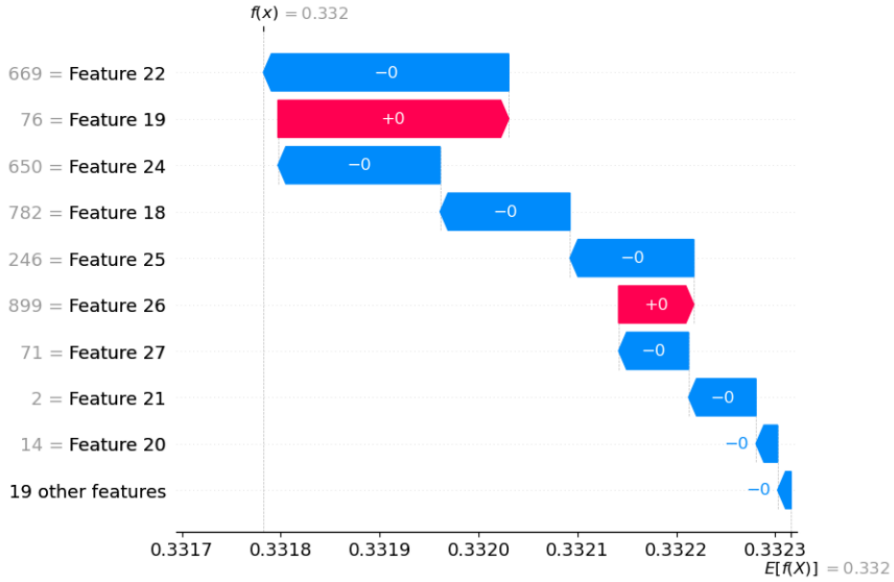
Mimo że wyniki analizy SHAP w tym przypadku nie wyszły zgodnie z oczekiwaniami, narzędzie SHAP zazwyczaj dostarcza wartościowych informacji na temat wpływu poszczególnych cech na predykcje modelu sentymentu. W prawidłowo przeprowadzonej analizie, SHAP powinien pokazywać, które słowa i frazy mają największy wpływ na końcowy wynik sentymentu, umożliwiając identyfikację cech przyczyniających się zarówno do pozytywnego, jak i negatywnego sentymentu. Wykresy SHAP dostarczają wizualnych interpretacji tych wpływów, co zazwyczaj pomaga lepiej zrozumieć działanie modelu i dokonać jego optymalizacji. W naszym przypadku chcieliśmy pokazać między innymi wykresy *waterfall*, pokazujące moc wpływu słów w konkretnej sekwencji na wybranie sentymentu oraz wykresy *beeswarm* pokazujące jak mocny wpływ miały konkretne słowa na generalizację sekwencji.



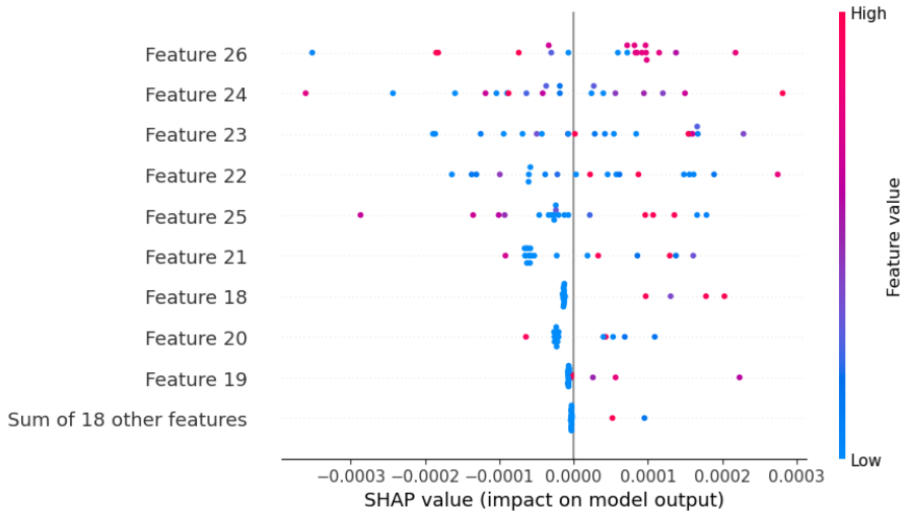
Analiza wartefall dla sentymentu negatywnego - featury to poszczególne słowa



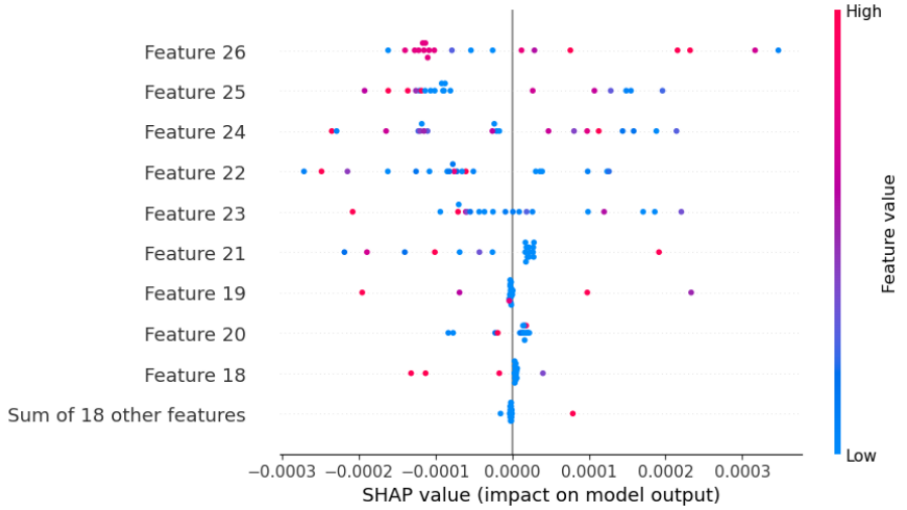
Analiza wartefall dla sentymentu neutralnego - featury to poszczególne słowa



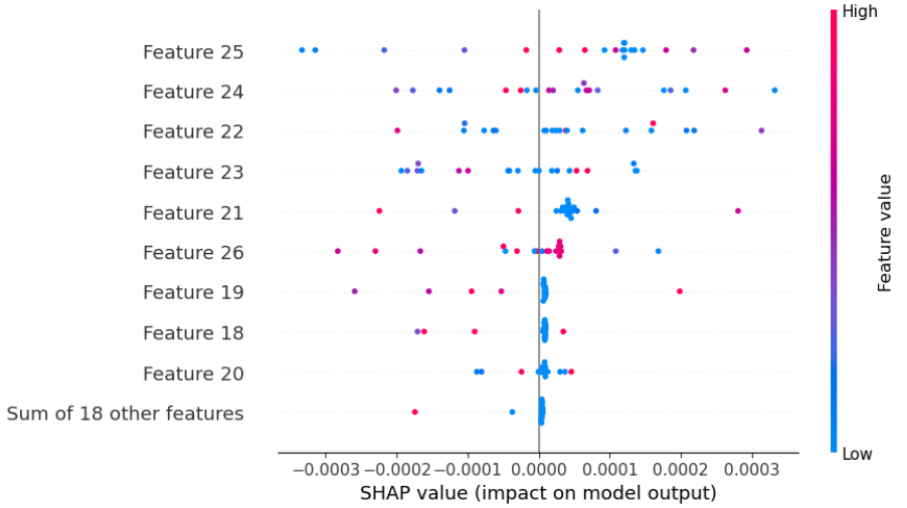
Analiza wartefall dla sentymentu pozytywnego - featury to poszczególne słowa



Analiza beeswarm dla sentymentu negatywnego - featury to poszczególne słowa



Analiza beeswarm dla sentymentu neutralnego - featury to poszczególne słowa



Analiza beeswarm dla sentymentu pozytywnego - featury to poszczególne słowa

6. Wnioski

- Analiza SHAP jest skutecznym narzędziem do wyjaśniania wyników modeli NLP, zwłaszcza w kontekście analizy sentymentu.
- Pozwala ona na zidentyfikowanie kluczowych cech wpływających na predykcje modelu, co może być pomocne w dalszym doskonaleniu modelu oraz zrozumieniu jego ograniczeń.
- Dzięki analizie SHAP możemy lepiej zrozumieć, jak model NLP interpretuje teksty i jakie elementy są dla niego najważniejsze w kontekście oceny sentymentu.
- Pomimo że w tym przypadku wyniki analizy SHAP nie spełniły oczekiwań, narzędzie to zwykle dostarcza wartościowych informacji na temat wpływu poszczególnych cech na predykcje sentymentu.