

[← Select Another Notebook/Commit/PR](#)

piotr



GitHub

Presentation.ipynb



Last edited by Piotr Krawiec 11 minutes ago



You're not receiving email notifications for any discussion on this notebook.

DISCUSSION

NOTEBOOK

CHANGES 7

Log-normal people

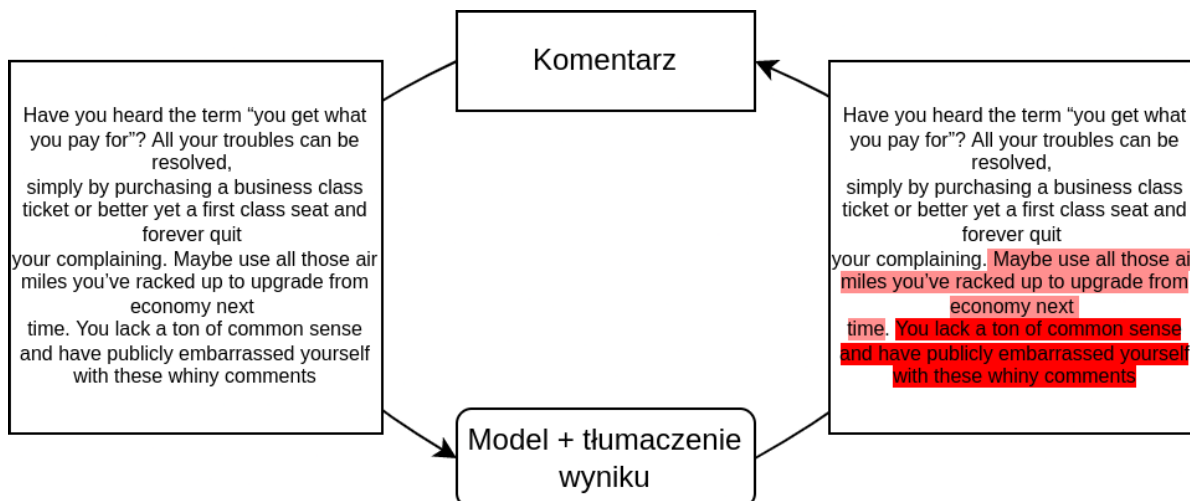
```
1 import transformers
2 import shap
3 import numpy
4 import matplotlib.pyplot as plt
5 import pandas as pd
6 import numpy as np
7 import scipy
8 from transformers import pipeline
9 from transformers import AutoTokenizer, AutoModelForSequenceClassification
10 import torch
```

Opis rozwiązania

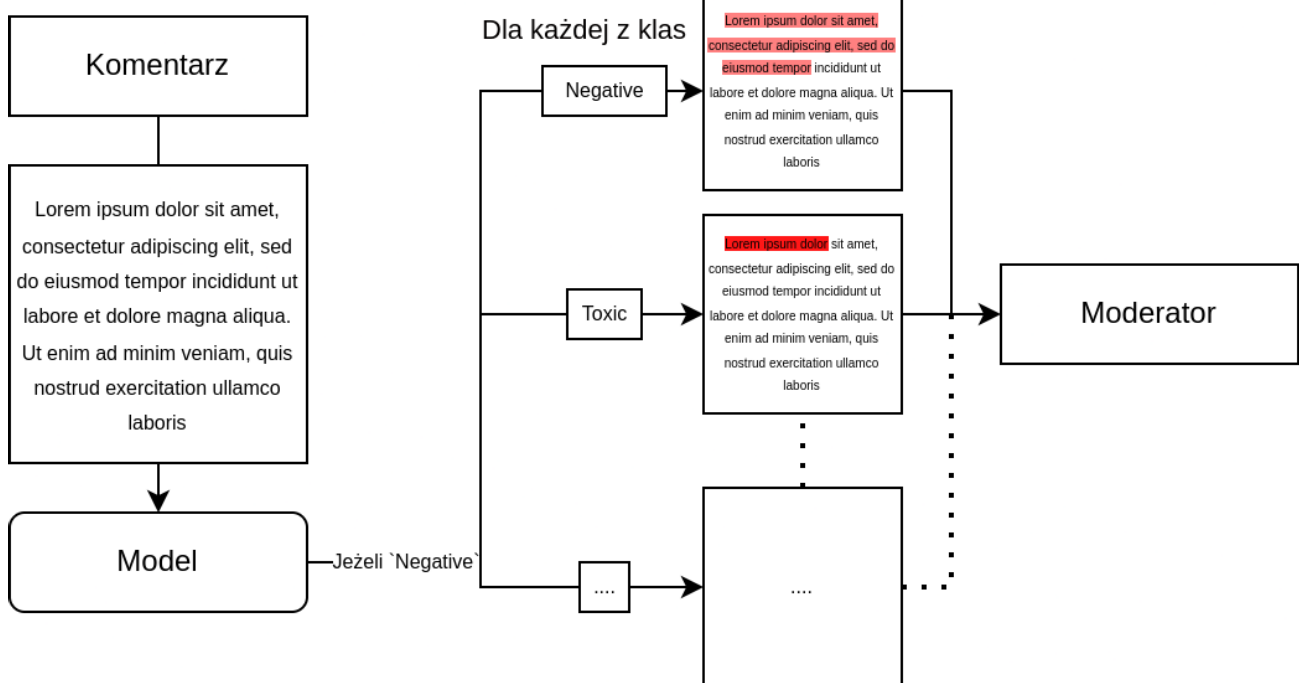
Celem projektu jest Nasze rozwiązanie składa się z 3 modułów. Dwa z nich to modele, które pozwalają klasyfikować tekst. Pierwszy przewiduje sentyment, a drugi rodzaj negatywnego nacechowania. A trzeci służy do objaśniania predykcji modelu. Modele te pobrane zostały ze strony Huggingface.io, ponieważ znajdują się tam gotowe wytrenowane modele, na których temat napisano prace naukowe i ich dokładność w różnych zadaniach mamy już sprawdzoną. Jest to bardzo ważne, gdyż od decyzji, które wygeneruje model zależy jakość moderacji.

Moduły można łączyć na kilka sposobów, które znajdują zastosowanie w wielu produktach takich jak:

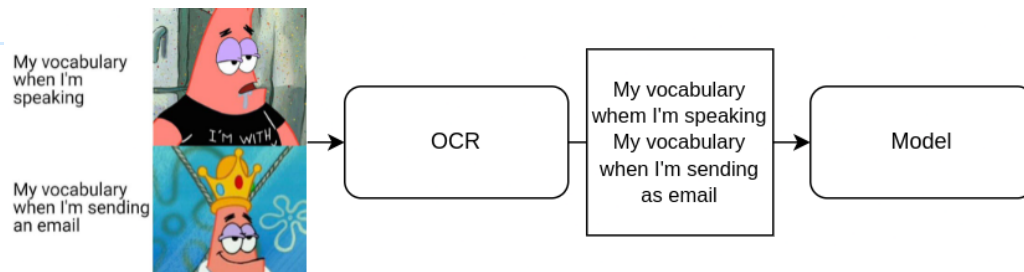
1. Analiza sentymentu po stronie użytkownika. Wpisując komentarz otrzymujemy informację czy nie jest on nacechowany negatywnie (toksyczność, groźba, dyskryminacja grup). Jeżeli jest, to nasz moduł objaśniania wskaże dla każdej z negatywnych cech miejsce jej występowania w komentarzu, które **użytkownik będzie mógł poprawić**. Z produktem poprawiającym negatywne nacechowanie użytkownika po jego stronie, jeszcze się nie spotkaliśmy.



2. Automatyczne zgłaszanie negatywnych komentarzy. Po wykryciu negatywnego sentymentu moderator otrzymuje komentarz z oznaczonym negatywnym fragmentem, co przyspiesza i ułatwia moderację. Szczególnie w przypadku długich tekstów, ponieważ **natychmiast wyróżnione są interesujące fragmenty**. W naszym rozwiązaniu chcemy, aby ostateczną decyzję zawsze podejmował moderator, nasz model ma mu wyłącznie ułatwić pracę. Żadne z dostępnych na rynku narzędzi nie pozwala na tak dokładne wyróżnienie wybranej negatywnej klasy. OpenAI pozwala na oznaczenie pozytywny/negatywny, podobnie wygląda to na Monkeylearn. Możemy "tłumaczyć" wybór dowolnej z negatywnych klas, nawet po zmianie modelu np. na taki który posiada więcej klas negatywnych.



Ponadto to rozwiązanie można rozszerzyć np. o inne media takie jak video czy audio, dla których tekst możemy otrzymać z pomocą transkrypcji video.



Moduł objaśniania modeli

Jest to nasz najważniejszy moduł, korzystając z biblioteki SHAP można objaśniać dowolne modele, także te tekstowe. Poniżej znajduje się przykład z biblioteki shap, w kolejnych rozdziałach pokażemy zastosowanie tego modułu we współpracy z naszymi modelami.

0th instance:



Tutaj dla każdego fragmentu tekstu, oznaczone mamy jaki miał wpływ na wyjście modelu. Dzięki temu jesteśmy w stanie dla wybranego negatywnego zachowania określić miejsce jego występowania w tekście.

Moduł badania sentymentu

Moduł ten odpowiada za badanie sentymentu, klasyfikuje dowolny tekst do jednej z 3 klas:

- Negative
- Neutral
- Positive

```

1 sentiment_pipe = pipeline("text-classification", model="cardiffnlp/twitter-roberta-base-sentiment")
2
3 def translate(label):
4     t = {"LABEL_0": "Negative",
5         "LABEL_1": "Neutral",
6         "LABEL_2": "Positive"}
7     return t[label]
  
```

```

1 def predict_sentiment(data):
2     label = translate(sentiment_pipe(data)[0]["label"])
3     return {"label": label, "score": sentiment_pipe(data)[0]["score"]}
  
```

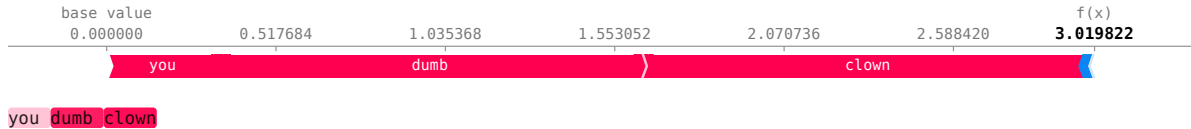
Przykład użycia

```
1 text = "you dumb clown"
2 predict_sentiment(text)
{'label': 'Negative', 'score': 0.9534616470336914}
```

Mając negatywny wynik możemy z pomocą Modułu objaśniania modeli znaleźć co sprawiło, że model oznaczył ten tekst jako Negative .

```
1 explainer = shap.Explainer(sentiment_pipe) # Moduł objaśniania
2 shap_values = explainer([text])
3 label = sentiment_pipe(text)[0]["label"]
4 shap.plots.text(shap_values[:, :, label]) # Wybieramy klasę objaśnianą (klasa )
```

0th instance:

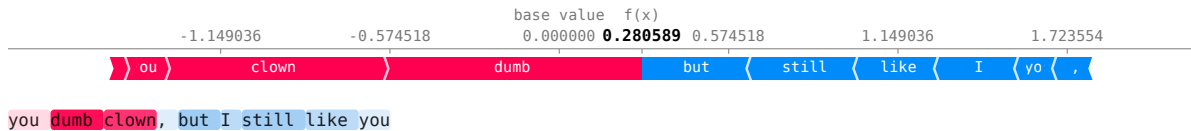


W skali od niebieskiego do czerwonego mamy pokazane, które fragmenty tekstu zadecydowały o wyniku modelu. Tutaj słowa dumb oraz clown zadecydowały o klasie Negative . Jeżeli zmienimy tekst, sprawimy, że będzie bardziej pozytywny.

```
1 text = "you dumb clown, but I still like you"
```

```
1 print(predict_sentiment(text))
2 shap_values = explainer([text])
3 label = sentiment_pipe(text)[0]["label"]
4 shap.plots.text(shap_values[:, :, label]) # Wybieramy klasę objaśnianą
{'label': 'Negative', 'score': 0.5696905851364136}
```

0th instance:



Sentyment nadal jest Negative , ale widzimy, że moduł objaśniający pokazuje nam, które słowa zmniejszyły prawdopodobieństwo tej klasy.

Moduł klasyfikacji negatywnego tekstu

Dla dowolnego tekstu oprócz tego, że tekst jest negatywny możemy określić dlaczego jest negatywny tj. wskazać prawdopodobieństwo należenia do każdej z klas:

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

Wracając do poprzedniego przykładu, obliczymy te prawdopodobieństwa.

```
1 # Tworzenie modelu klasyfikacji negatywnego tekstu
2 clf_tokenizer = AutoTokenizer.from_pretrained("unitary/toxic-bert")
3 clf_model = AutoModelForSequenceClassification.from_pretrained("unitary/toxic-bert")
4 clf_pred = transformers.pipeline("text-classification", model=clf_model, tokenizer=clf_tokenizer,
return_all_scores=True)
```

```
1 text = "you dumb clown" # Przykład
2 clf_pred(text)
[[{'label': 'toxic', 'score': 0.7617954015731812},
 {'label': 'severe_toxic', 'score': 0.00035054481122642756},
 {'label': 'obscene', 'score': 0.014293862506747246},
 {'label': 'threat', 'score': 1.986711322388146e-05},
 {'label': 'insult', 'score': 0.22305439412593842},
 {'label': 'identity_hate', 'score': 0.0004858026804868132}]]
```

Najbardziej prawdopodobne jest, że tekst ten zawiera klasy: `toxic` oraz `insult`. Poniżej objaśnię dlaczego model z dużym prawdopodobieństwem przypisał klasę `toxic`:

```
1 clf_explainer = shap.Explainer(clf_pred)
2 shapley_values = clf_explainer(np.array([text]))
3 shap.plots.text(shapley_values[:, :, 'toxic']) # Klasa toxic
```

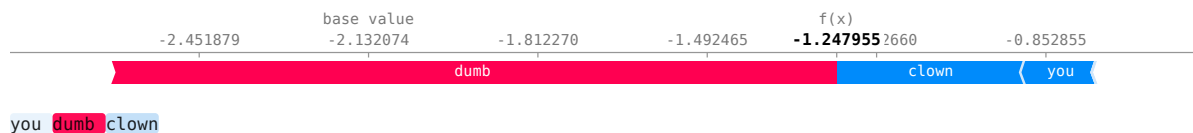
0th instance:



Oraz klasę `insult`:

```
1 shap.plots.text(shapley_values[:, :, 'insult']) # Klasa insult
```

0th instance:



Przykłady

Pozytywny tekst

Jak dotąd wszystkie zaprezentowane przykłady były negatywne, poniżej zaprezentowałem jak model reaguje na pozytywne komentarze.

```
1 text = "I really enjoyed your work, keep it up"
2 print(predict_sentiment(text))
{'label': 'Positive', 'score': 0.9836942553520203}
```

```
1 clf_pred(text)
[{'label': 'toxic', 'score': 0.0007015864248387516},
 {'label': 'severe_toxic', 'score': 0.00012133460404584184},
 {'label': 'obscene', 'score': 0.00017723572091199458},
 {'label': 'threat', 'score': 0.00014815809845458716},
 {'label': 'insult', 'score': 0.0001734427351038903},
 {'label': 'identity_hate', 'score': 0.00014075862418394536}]
```

Długi tekst

```
1 text = """Have you heard the term "you get what you pay for"? All your troubles can be resolved,
2 simply by purchasing a business class ticket or better yet a first class seat and forever quit
3 your complaining. Maybe use all those air miles you've racked up to upgrade from economy next
4 time. You lack a ton of common sense and have publicly embarrassed yourself with these whiny comments."""
```

```
1 predict_sentiment(text)
{'label': 'Negative', 'score': 0.868039608001709}
```

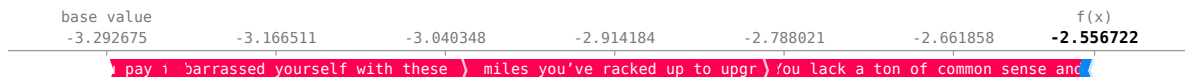
```
1 clf_pred(text)
[{'label': 'toxic', 'score': 0.9142789840698242},
 {'label': 'severe_toxic', 'score': 0.0007636288064531982},
 {'label': 'obscene', 'score': 0.008510921150445938},
 {'label': 'threat', 'score': 0.0014207024360075593},
 {'label': 'insult', 'score': 0.07197616994380951},
 {'label': 'identity_hate', 'score': 0.0030496353283524513}]
```

```

1 shapley_values = clf_explainer(np.array([text]))
2 shap.plots.text(shapley_values[:, :, 'insult']) # Klasa insult
    0%|          | 0/48 [00:00<?, ?it/s]

```

0th instance:



Have you heard the term "you get what you pay for"? All your troubles can be resolved, simply by purchasing a business class ticket or better yet a first class seat and forever quit your complaining. Maybe use all those air miles you've racked up to upgrade from economy next time. You lack a ton of common sense and have publicly embarrassed yourself with these whiny comments.

Model działa również dla długich tekstów, poprawnie rozpoznał iż komentarz był negatywnie nastawiony oraz toksyczny. Ponadto dokładnie wskazał miejsce, w którym autor artykułu został urażony.

Komentarze z Reddita

```

1 import praw
2
3 reddit = praw.Reddit(client_id="NivQ32fjBvLnDcIMXR70Kw",
4                       client_secret="xBewFjWNDf78Rj_UeZk605eZlHe_yg",
5                       user_agent="ABC")
6
7
8 submission = reddit.submission(url="https://www.reddit.com/r/canada/comments/s46fmw
9 /harry_rakowski_if_we_tax_the_unvaccinated_what/")
10 submission.comments.replace_more(limit=10)
11 for comment in submission.comments.list():
12     print((comment.body).replace(">", ""))

```

```

1 text = """
2 We should tax the s**t out of soft drinks and other fast foods and snacks. Sodium and sugar, coupled with a
3 sedentary lifestyle are the true killers.
4 How ironic that obesity is one of the leading comorbidities associated with having a bad run at nearly every kind
5 of illness.
6 """

```

```

1 predict_sentiment(text)
{'label': 'Negative', 'score': 0.9535958766937256}

```

```

1 clf_pred(text)
[[{'label': 'toxic', 'score': 0.7127734422683716},
 {'label': 'severe_toxic', 'score': 0.0027107808273285627},
 {'label': 'obscene', 'score': 0.19764475524425507},
 {'label': 'threat', 'score': 0.0018170236144214869},
 {'label': 'insult', 'score': 0.07720550149679184},
 {'label': 'identity_hate', 'score': 0.00784845370799303}]]

```

Przedstawię wykres dla klasy `obscene`, ponieważ dla klasy `toxic` prawdopodobieństwo jest relatywnie niskie.

```

1 shapley_values = clf_explainer(np.array([text]))
2 shap.plots.text(shapley_values[:, :, 'obscene']) # Klasa obscene

```

0th instance:



We should tax the s**t out of soft drinks and other fast foods and snacks. Sodium and sugar, coupled with a sedentary lifestyle are the true killers. How ironic that obesity is one of the leading comorbidities associated with having a bad run at nearly every kind of illness.

Wszystkiwanie tekstu z obrazu

```
1 path = r"C:\\Users\\Admin\\Desktop\\BITEhack\\BITEHack\\img\\test1.jpg"
```

```
1 import cv2
2 import pytesseract
3 import numpy as np
4 import re
5
6 pytesseract.pytesseract.tesseract_cmd = r"C:\\Program Files (x86)\\Tesseract-OCR\\tesseract"
7
8 # get grayscale image
9 def get_grayscale(image):
10     return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
11
12 #thresholding
13 def thresholding(image):
14     return cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]
15
16 img = cv2.imread(path)
17
18
19 gray = get_grayscale(img)
20 gray = thresholding(gray)
21
22
23 text = pytesseract.image_to_string(gray)
24 text = re.sub(' +', ' ', text.replace("\n", " "))
```

Teraz sprawdzamy czy skrypt prawidłowo odczytał tekst z obrazu

```
1 text
```

```
1 predict_sentiment(text)
{'label': 'Neutral', 'score': 0.8269897699356079}
```

Tekst został prawidłowo oceniony przez model jako nietoksyczny/nienacechowany negatywnymi emocjami

```
1 shapley_values = clf_explainer(np.array([text]))
2 clf_pred(text)
[{'label': 'toxic', 'score': 0.5706530809402466},
 {'label': 'severe_toxic', 'score': 0.061839502304792404},
 {'label': 'obscene', 'score': 0.10942654311656952},
 {'label': 'threat', 'score': 0.06952789425849915},
 {'label': 'insult', 'score': 0.10652746260166168},
 {'label': 'identity_hate', 'score': 0.08202551305294037}]]
```