Log-normal people

In [37]:
```
import transformers
import shap
import numpy
import matplotlib.pylab as plt
import pandas as pd
import numpy as np
import scipy
from transformers import pipeline
from transformers import AutoTokenizer, AutoModelForSequenceClassification
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 Hugginface.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 zależeć będzie jakość moderacji.

Moduły można łączyć na kilka sposobów, które znajdą 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 uzytkownika po jego stronie, jeszcze się nie spotkaliśmy.

1. 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 dostostępnych na rynku narzędzi nie pozwala na tak dokładne wróżnienie 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.

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 sentymenu, sklasyfikuje dowolny tekst do jednej z 3 klas:

- Negative
- Neutral
- Positive

In [38]:
```
sentiment_pipe = pipeline("text-classification", model="cardiffnlp/twitter-rober

def translate(label):
    t = {"LABEL_0": "Negative",
     "LABEL_1": "Neutral",
     "LABEL_2": "Positive"}
    return t[label]
```

In [39]:
```
def predict_sentiment(data):
    label = translate(sentiment_pipe(data)[0]["label"])
    return {"label": label, "score": sentiment_pipe(data)[0]["score"]}
```

Przykład użycia

In [40]:
```
text = "you dumb clown"
predict_sentiment(text)
```

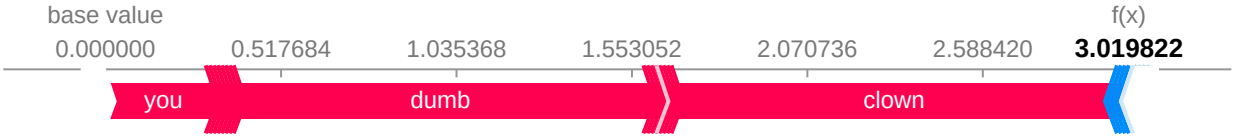
Out[40]:
```
{'label': 'Negative', 'score': 0.9534616470336914}
```

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

In [41]:
```
explainer = shap.Explainer(sentiment_pipe) # Moduł objaśniania
shap_values = explainer([text])
```

```
label = sentiment_pipe(text)[0]["label"]
shap.plots.text(shap_values[:,:, label]) # Wybieramy klasę objaśnianą (klasa )
```

0th instance:



you dumb clown

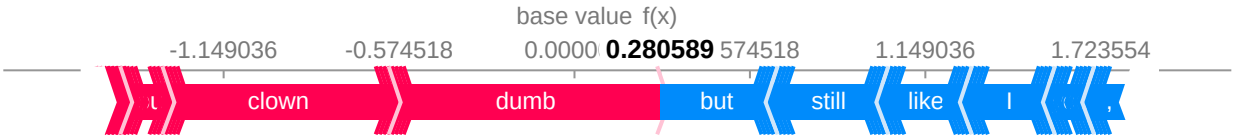
W skali od niebieskiego do czerwonego mamy pokazane, które fragmenty tekstu zadecyowały o wyniku modelu. Tutaj słowa `dumb` oraz `clown` zadecydowały o klasie `Negative` . Jeżeli zmienimy tekst, sprawimy, że będzie bardziej pozytywny.

In [42]:
```
text = "you dumb clown, but I still like you"
```

In [43]:
```
print(predict_sentiment(text))
shap_values = explainer([text])
label = sentiment_pipe(text)[0]["label"]
shap.plots.text(shap_values[:,:, label]) # Wybieramy klasę objaśnianą
```

```
{'label': 'Negative', 'score': 0.5696905851364136}
```

0th instance:



you dumb clown, but I still like you

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.

In [44]:
```
# Tworzenie modelu klasyfikacji negatywnego tekstu
clf_tokenizer = AutoTokenizer.from_pretrained("unitary/toxic-bert")
clf_model = AutoModelForSequenceClassification.from_pretrained("unitary/toxic-be
clf_pred = transformers.pipeline("text-classification", model=clf_model, tokeniz
```

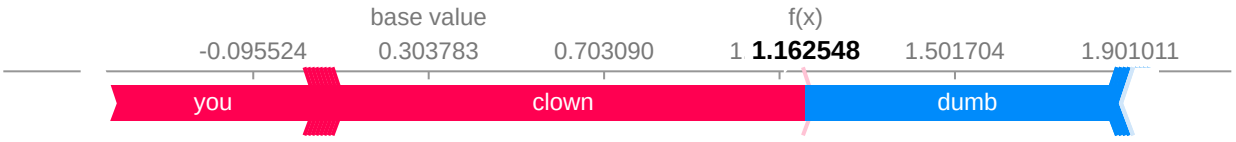
In [50]:
```
text = "you dumb clown" # Przykład
clf_pred(text)
```

Out[50]:
```
[[{'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 objaśniłem dlaczgo model z dużym prawdopodobieństwem przypisał klasę `toxic` :

In [53]:
```
clf_explainer = shap.Explainer(clf_pred)
shapley_values = clf_explainer(np.array([text]))
shap.plots.text(shapley_values[:, :, 'toxic']) # Klasa toxic
```

0th instance:

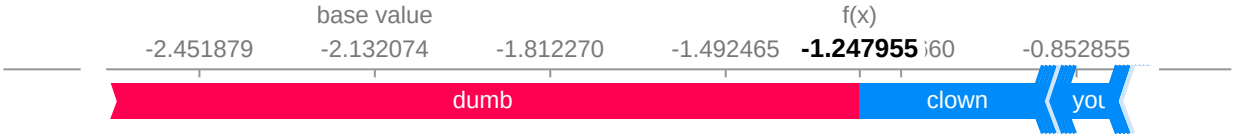


you dumb clown

Oraz klasę `insult` :

In [54]:
```
shap.plots.text(shapley_values[:, :, 'insult']) # Klasa insult
```

0th instance:



you dumb clown

Przykłady

Pozytywny tekst

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

In [12]:
```
text = "I really enjoyed your work, keep it up"
print(predict_sentiment(text))
```

```
{'label': 'Positive', 'score': 0.9836942553520203}
```

In [13]:
```
clf_pred(text)
```

Out[13]:
```
[[{'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

In [55]:
```
text = """Have you heard the term “you get what you pay for”? All your troubles
simply by purchasing a business class ticket or better yet a first class seat ar
your complaining. Maybe use all those air miles you’ve racked up to upgrade from
time. You lack a ton of common sense and have publicly embarrassed yourself with
```

In [56]:
```
predict_sentiment(text)
```

Out[56]:
```
{'label': 'Negative', 'score': 0.868039608001709}
```

In [57]:
```
clf_pred(text)
```

Out[57]:
```
[[{'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': 0.003049635328352451 3}]]
```

In [58]:
```
shapley_values = clf_explainer(np.array([text]))
shap.plots.text(shapley_values[:, :, 'insult']) # Klasa insult
```

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 komentaż był negatywnie nastawiony oraz toksyczny. Ponadto dokładnie wskazał mniejsce, w którym autor artykułu został

urażony.

Komentarze z Reddita

In [ ]:
```
import praw

reddit = praw.Reddit(client_id="NivQ32fjBvLnDc1MXR70Kw",
                     client_secret="xBewFjWNDf78Rj_UeZk6O5eZlHe_yg",
                     user_agent="ABC")


submission = reddit.submission(url="https://www.reddit.com/r/canada/comments/s40
submission.comments.replace_more(limit=10)
for comment in submission.comments.list():
    print((comment.body).replace(">",""))
```

In [59]:
```
text = """
We should tax the s**t out of soft drinks and other fast foods and snacks. Sodiu
How ironic that obesity is one of the leading comorbidities associated with havi
"""
```

In [60]:
```
predict_sentiment(text)
```

Out[60]:
```
{'label': 'Negative', 'score': 0.9535958766937256}
```

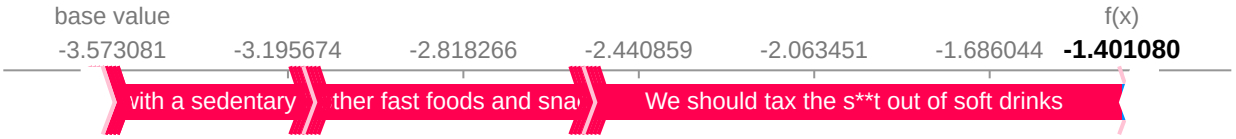
In [61]:
```
clf_pred(text)
```

Out[61]:
```
[[{'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': 0.007848453707993033}]]
```

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

In [62]:
```
shapley_values = clf_explainer(np.array([text]))
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.

Wszystywanie tekstu z obrazu

In [31]:
```
path = r"C:\\Users\\Admin\\Desktop\\BITEhack\\BITEHack\\img\\test1.jpg"
```

In [36]:
```
import cv2
import pytesseract
import numpy as np
import re

pytesseract.pytesseract.tesseract_cmd = r"C:\\Program Files (x86)\\Tesseract-OCR

# get grayscale image
def get_grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#thresholding
def thresholding(image):
    return cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

img = cv2.imread(path)


gray = get_grayscale(img)
gray = thresholding(gray)


text = pytesseract.image_to_string(gray)
text = re.sub(' +', ' ', text.replace("\n"," "))
```

Teraz sprawdzamy czy skrypt prawidłowo odczytał tekst z obrazu

In [63]:
```
text
```

In [64]:
```
predict_sentiment(text)
```

Out[64]:
```
{'label': 'Neutral', 'score': 0.8269897699356079}
```

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

In [65]:
```
shapley_values = clf_explainer(np.array([text]))
clf_pred(text)
```

Out[65]:
```
[[{'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}]]
```