Projekt Metody Inteligencji Obliczeniowej

Temat: Predykcja zainteresowania postami w social media z użyciem metod NLP

Bartosz Bardian, Jędrzej Szostak

1. Opis projektu

Na podstawie zbioru tweetów Donalda Trumpa zebranych w zbiorze:
https://www.kaggle.com/datasets/austinreese/trump-tweets
zbudowaliśmy modele predykcyjne służący do analizy popularności tweetów (w formie retweetów) stosując reprezentację występujących w tekście słów. Następnie dla zbudowanego modelu przeprowadziliśmy analizę SHAP. Pierwszy z modeli wykorzystuje regresję liniową, a drugi sieć neuronową. Projekt pozwala również na napisanie własnego tweetu i ocenienie jaką będzie miał popularność (zakładając, że jesteśmy Donaldem Trumpem).
Wyniki przedstawiliśmy w formie wykresów i map słów oraz porównaliśmy modele ze sobą.

2. Kod - architektura i opis

Wykorzystanie bibliotek:

- re - wykorzystanie wyrażeń regularnych do usuwania znaków z bazy
- pandas - obsługa bazy tweetów
- random - losowe wartości, przemieszanie tablicy
- matplotlib.pyplot - rysowanie wykresów
- sklearn - wykorzystanie gotowych funkcji takich jak LinearRegression (regresja liniowa), MLPRegressor (regresja wielowarstwowym perceptronem), train_test_split (podział zbioru na uczący i testujący); wykorzystanie modułu text do usunięcia nic nieznaczących słów w języku angielskim (mogłyby zaburzać wyniki) oraz konwersja tablicy stringów do wektorów określających obecność słów związanych z danym indeksem w danym tweecie

Algorytm krok po kroku:

1. Importowanie bibliotek
2. Pobranie bazy
3. Wyczyszczenie bazy ze znaków specjalnych, linków i cyfr
4. Podzielenie zbioru na zbiór uczący i testowy
5. Wykorzystanie regresji liniowej do predykcji ze zbioru uczącego
6. Powtórzenie kroków 4-6 dla modelu MLPRegression
7. Zapisanie modeli
8. Sprawdzenie wyników i ich porównanie

Opis poszczególnych funkcji i fragmentów kodu:

- get_data_from_file(filename) - podajemy nazwę pliku csv, konwertujemy jego zawartość do obiektu DataFrame, dzielimy zbiór na dwie listy, jedna zawierająca treść tweeta, a druga zawierająca liczbę retweetów i zwracamy obie te listy

- clean_tweets(tweets) - jako parametr przyjmujemy listę z zawartością tweetów, następnie usuwamy z niej zbędne słowa, linki, godziny, liczby i ją zwracamy

- draw_chart(retweets) - jako parametr przyjmujemy liczbę retweetów dla kolejnych tweetów i rysujemy słupkowy wykres ich popularności

- vectorize_tweets(tweets,retweets,n_occurences=50,own_stop_words=[]) - jako parametry przyjmujemy kolejno listę z zawartością tweetów, listę z wartością retweetów dla poszczególnych tweetów, minimalna liczba wystąpień słowa w całym zbiorze (aby odseparować szum wynikający z bardzo rzadkich słów), listę słów takich jak “th” czy “nd”, ktore nic nie znaczą; następnie wykluczamy kolejne słowa ze zbioru (takie z wartością wystąpień mniejsza niż ta przekazana w parametrze n_occurences oraz takie które zawierać się będą w liście own_stop_words), zamieniamy listę tweetów na zbiór wektorów oraz normalizujemy liczbę retweetów tak, aby ich wartość zawierała się w przedziale [0, 1], zwracamy zbiór wektorów tweetów i znormalizowaną listę retweetów

- get_words(vectorizer) - jako parametr przyjmuje zbiór wektorów i zwraca listę wszystkich występujących w nich słów

- get_linear_regression_model(X_train, y_train) - jako parametry podajemy wcześniej otrzymane z funkcji train_test_split zbiory uczące, tworzymy model regresji liniowej i uczymy go przy pomocy zbiorów, następnie zwracamy ten model

- check_improvement(y_test, y_pred) - jako parametry przyjmujemy odpowiednio zbiór testowy liczby retweetów oraz odpowiadający mu zbiór przewidywanej ilości retweetów przez nasz model, błąd obliczamy jako różnicę pomiędzy wartościami ze zbioru testowego i otrzymanymi z modelu, następnie zapisujemy losowy błąd jaki wyszedłby w

przypadku losowego dobrania tweetów ze zbioru testującego z wartościami retweetów

- get_shap(model, all_words) - jako parametry przyjmujemy odpowiednio nasz model i listę wszystkich słów, a następnie tworzymy wektory zawierające pojedyncze słowa (odpowiednik jednosłownych tweetów). Dla każdego z nich dokonujemy predykcji i zapisujemy wynik. Zwracamy listę słów wraz z wynikiem predykcji ich popularności.

- get_mlp_regression_model(X_train, y_train, epochs, layers) - jako parametry przyjmujemy odpowiednio nasze zbiory uczące X_train, y_train, liczbę epok i warstw jakie chcemy żeby nasz model posiadał, następnie przy pomocy funkcji MLPRegression uczymy nasz model przy pomocy zbiorów oraz wybranej liczby epok i warstw, na koniec zwracamy model

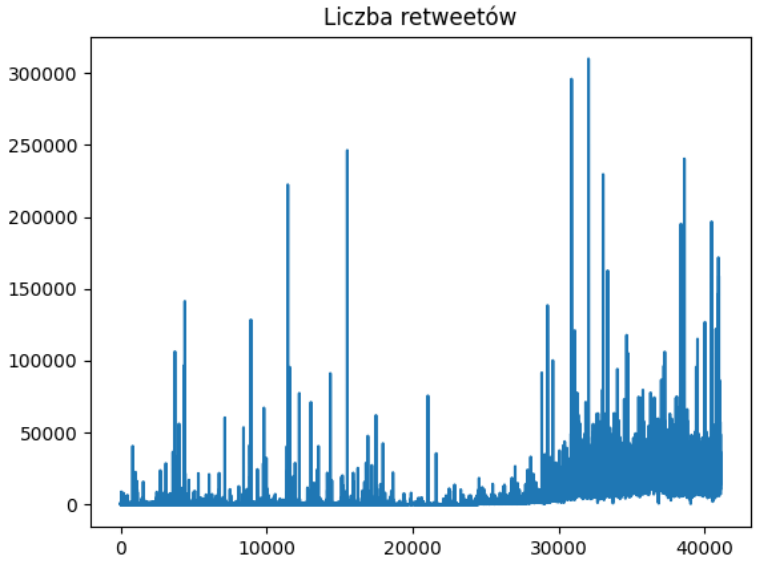
- draw_cloud(shap) - na podstawie przeprowadzonej analizy popularności poszczególnych słów rysujemy mapę słów, gdzie wielkość odzwierciedla jego wpływ na wartość predykcji (popularność)

- save_model(model, filename) - zapisujemy wytrenowany model przy pomocy biblioteki pickle w pliku o podanej nazwie i rozszerzeniu ".sav", po to by móc go wykorzystać bez ponownego trenowania

- load_model(filename) - zwracamy model zapisany uprzednio przy pomocy funkcji save_model w pliku o rozszerzeniu ".sav"

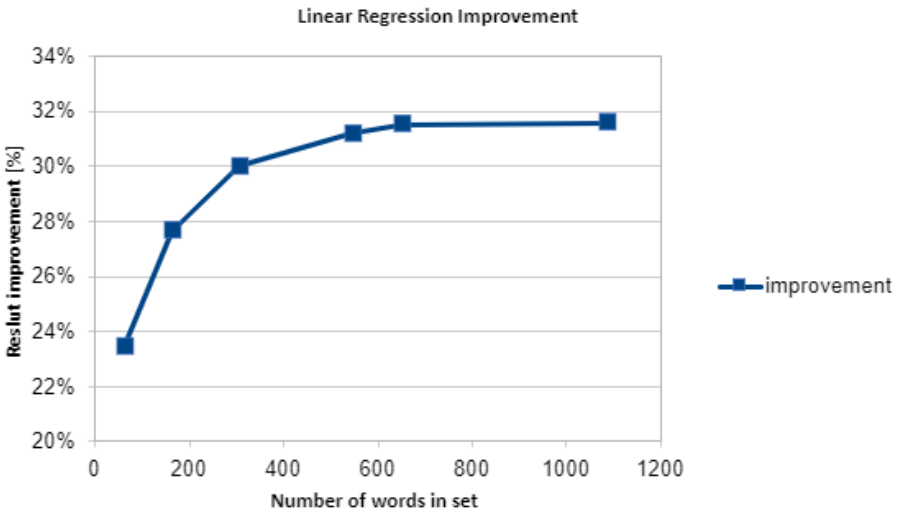
3. Wyniki

Na początku zaprezentowaliśmy popularność tweetów Donalda Trumpa na przestrzeni czasu:



Rys 1. Liczba retweetów Donalda Trumpa w zależności od ilości tweetów już przez niego wypuszczonych.

Następnie dla regresji liniowej staraliśmy znaleźć się optymalne rozwiązanie, aby tego dokonać manipulowaliśmy liczbą słów, która wchodziła do zbioru poprzez ograniczanie lub zwiększanie liczby minimalnej ilości wystąpień słów na przestrzeni wszystkich tweetów z bazy. Wyniki prezentują się następująco:

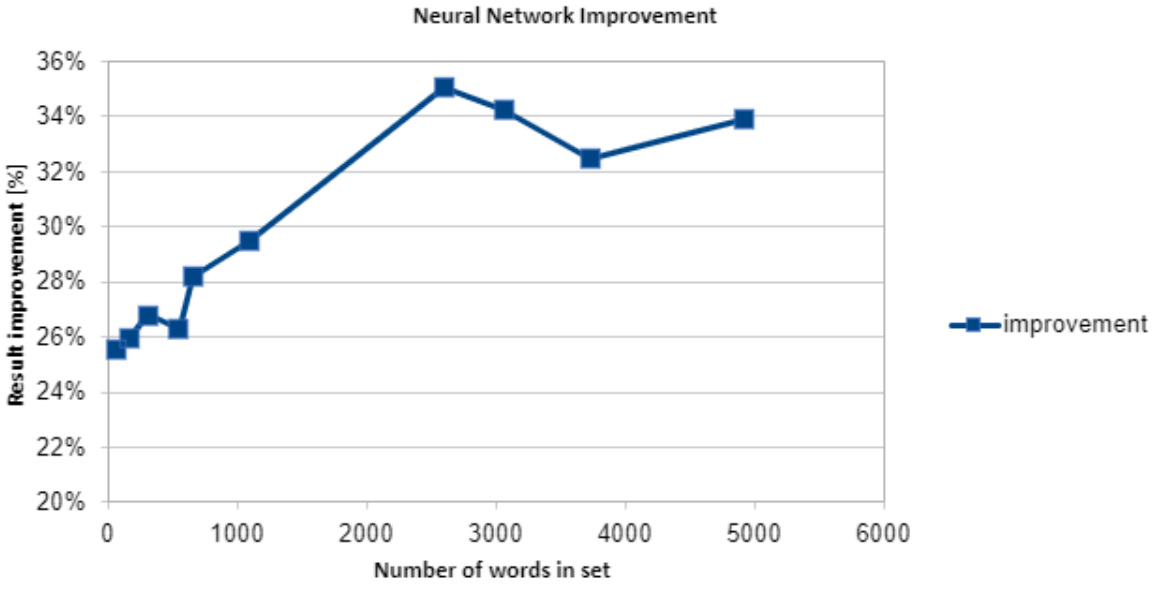


Rys 2. Poprawa wyników dla zwiększającej się liczby słów w zbiorze dla regresji liniowej.

Jak widzimy dokładność dopasowania wzrasta wraz z liczbą słów w zbiorze, ale po osiągnięciu pewnego pułapu wartość stabilizuje się. Dzięki temu możemy ograniczyć liczbę słów co pozwoli przyspieszyć cały proces. Po wykonaniu modelu wykonaliśmy chmurę słów, które miały największy wpływ na popularność tweetów. Wyniki prezentują się następująco:

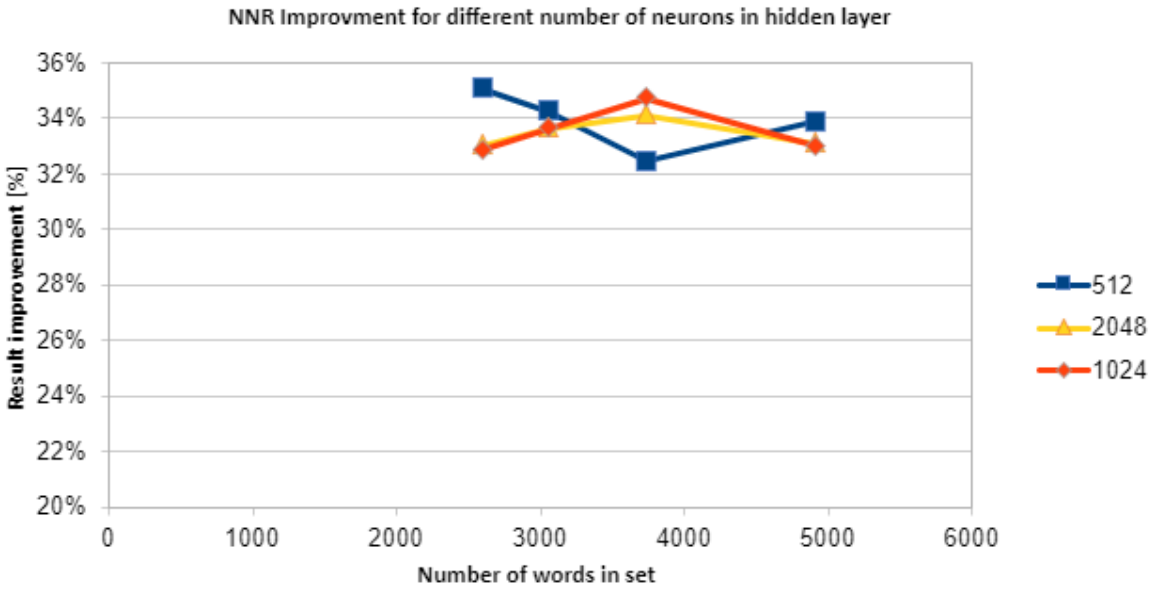
Rys 3. Chmura "najpopularniejszych" słów dla regresji liniowej.

Następnie przeszliśmy do wykonania tego zadania przy pomocy sieci neuronowej. Stworzyliśmy kolejny model i tutaj również sprawdziliśmy dla jakiej liczby słów wyniki są najlepsze:



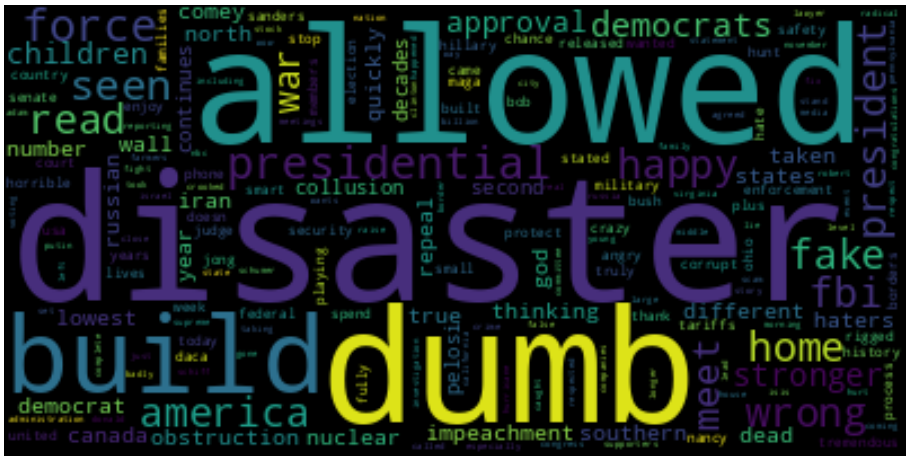
Rys 4. Poprawa wyników dla zwiększającej się liczby słów w zbiorze dla sieci neuronowej.

Jak widzimy na powyższym wykresie, model ten lepiej radzi sobie dla większej ilości słów niż w przypadku regresji liniowej, osiągając maksimum dla 2599. Dlatego też, dla tej liczby słów kontynuowaliśmy badanie modelu. Następnym krokiem było ustalenie optymalnej liczby neuronów w warstwie ukrytej (badanie pokazało, że większa ilość warstw ukrytych nie daje znacznie lepszych rezultatów, zaś komplikuje model wydłużając czas jego uczenia).
Okazało się, że optymalnym modelem jest ten z warstwą ukrytą z 512 neuronami.



Rys 5. Rezultaty dla różnych liczb neuronów w warstwie ukrytej.

Wykonaliśmy również analogicznie wykres chmury słów dla wyników otrzymanych dzięki modelowi opartemu na sieci neuronowej:



Rys 6. Chmura słów dla modelu sieci neuronowej.

Jak widać, słowa te różnią się od tych, które miały największy wpływ na popularność w przypadku modelu regresji liniowej. Analiza na przykładzie tweetów ze zbioru testowego oraz własnych pokazuje jednak, że oba modele podobnie radzą sobie z oceną popularności. Nawet jeśli wartość predykcji jest różna to kolejność tweetów wg popularności jest zazwyczaj bardzo podobna.

4. Źródło

https://github.com/bartekx43/Tweet_popularity_analysis/