Budowa modelu wykrywającego fake newsy

Adam Paleczny, Jarosław Skwarczek, Mikołaj Słowikowski

Czerwiec 2024

1 Opis

Celem tego projektu jest opracowanie modelu uczenia maszynowego, który potrafi dokładnie wykrywać fałszywe artykuły prasowe. Fałszywe wiadomości są poważnym problemem w dzisiejszej erze cyfrowej, ponieważ mogą rozpowszechniać dezinformację i powodować szkody. Nasz projekt ma na celu rozwiązanie tego problemu poprzez stworzenie niezawodnego i skutecznego systemu wykrywania fałszywych wiadomości.

2 Wstęp teoretyczny

W projekcie wykorzystano różne techniki klasyfikacji tekstu, takie jak drzewa decyzyjne, regresję logistyczną oraz sieci neuronowe typu Long Short-Term Memory (LSTM). Każda z tych metod posiada swoje unikalne zalety i ograniczenia, które wpływają na skuteczność detekcji fałszywych informacji.

2.1 Drzewa Decyzyjne

Drzewa decyzyjne są jedną z podstawowych metod klasyfikacji w uczeniu maszynowym. Polegają one na tworzeniu modelu predykcyjnego w postaci drzewa, gdzie każdy węzeł reprezentuje decyzję na podstawie wartości atrybutu, a gałęzie - wynik tej decyzji. Drzewa decyzyjne są łatwe do interpretacji i stosunkowo szybkie w trenowaniu, jednak mogą być podatne na przeuczenie (overfitting).

2.2 Regresja Logistyczna

Regresja logistyczna jest techniką statystyczną używaną do klasyfikacji binarnej, która szacuje prawdopodobieństwo przynależności obserwacji do jednej z dwóch klas. Model ten jest prosty do implementacji i interpretacji, jednak jego skuteczność może być ograniczona w przypadku bardziej złożonych wzorców w danych.

2.3 Sieci LSTM

Long Short-Term Memory (LSTM) to zaawansowany rodzaj rekurencyjnych sieci neuronowych (RNN), zaprojektowany do uczenia się długoterminowych zależności w sekwencjach danych. LSTM posiadają mechanizmy bramek (input, forget, output gate), które kontrolują przepływ informacji wewnątrz komórek, umożliwiając efektywne przechowywanie i odświeżanie stanu wewnętrznego sieci. Dzięki temu, LSTM są szczególnie skuteczne w przetwarzaniu i analizie sekwencji tekstowych, co sprawia, że są idealnym narzędziem do wykrywania fake news.

Bidirectional LSTM to rozszerzenie standardowej LSTM, które pozwala na przetwarzanie sekwencji danych w obu kierunkach – od początku do końca i od końca do początku. Dzięki temu model może lepiej uchwycić kontekst, który może pochodzić zarówno z wcześniejszych, jak i późniejszych słów w sekwencji.

3 Dane wejściowe

W projekcie wykorzystano zbiór danych zawierający wypowiedzi polityczne, które są klasyfikowane na podstawie ich prawdziwości. Dane te pochodzą z różnych źródeł, w tym z mediów, transkrypcji przemówień i wywiadów. Każdy rekord w zbiorze danych zawiera kilka kluczowych atrybutów, które są wykorzystywane do analizy i klasyfikacji.

3.1 Struktura Danych

Zbiór danych składa się z następujących kolumn:

- **politicianID**: Unikalny identyfikator polityka, który wygłosił daną wypowiedź.
- **name**: Imię i nazwisko polityka.
- **party**: Partia polityczna, do której należy polityk.
- **statementID**: Unikalny identyfikator wypowiedzi.
- **statementText**: Tekst wypowiedzi polityka, który jest przedmiotem analizy.
- **statementState**: Klasyfikacja wypowiedzi określająca jej prawdziwość. Wartości mogą być następujące:
  - TRUE: Wypowiedź jest prawdziwa.
  - FALSE: Wypowiedź jest fałszywa.
  - MISLEADING: Wypowiedź jest wprowadzająca w błąd.
  - UNVERIFIABLE: Wypowiedź jest niezweryfikowalna.

- **statementExplan**: (opcjonalnie) Dodatkowe wyjaśnienia dotyczące wypowiedzi, które mogą zawierać kontekst lub szczegółowe informacje wspierające klasyfikację.

3.2 Przetwarzanie Tekstu

W celu przygotowania danych do analizy, przeprowadzono szereg kroków przetwarzania wstępnego:

- **Czyszczenie Tekstu**: Usunięcie znaków interpunkcyjnych, konwersja tekstu na małe litery oraz usunięcie niepotrzebnych znaków.
- **Tokenizacja**: Podział tekstu na pojedyncze słowa (tokeny).
- **Usuwanie Stop Słów**: Usunięcie powszechnie występujących słów, które nie niosą ze sobą wartości informacyjnej (np. "i", "oraz", "na").
- **Konwersja do Liczb**: Zamiana słów na numeryczne reprezentacje za pomocą technik takich jak one-hot encoding czy TF-IDF (Term Frequency-Inverse Document Frequency).

3.3 Podział Danych

Zbiór danych został podzielony na zestawy treningowe, walidacyjne i testowe w celu stworzenia i oceny modeli klasyfikacyjnych. Dzięki temu możliwe było trenowanie modeli na jednej części danych i testowanie ich skuteczności na niezależnych zbiorach danych.

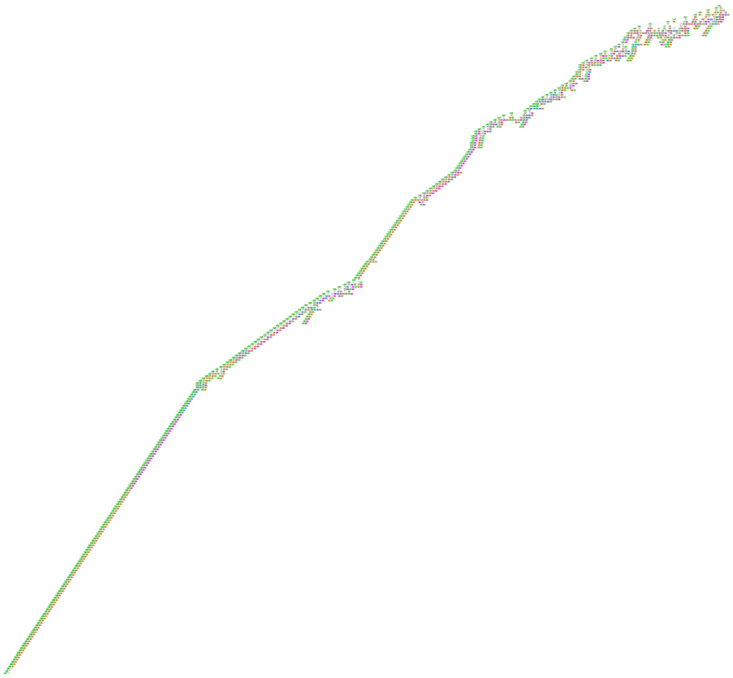
4 Drzewa decyzyjne

W projekcie zastosowano drzewo decyzyjne do klasyfikacji wypowiedzi politycznych na podstawie ich prawdziwości. Proces implementacji obejmował następujące kroki:

1. **Przygotowanie Danych**: Dane zostały wstępnie przetworzone, obejmując czyszczenie tekstu, tokenizację i konwersję do reprezentacji liczbowej za pomocą techniki TF-IDF.
2. **Podział Danych**: Zbiór danych został podzielony na zestawy treningowe i testowe.
3. **Trenowanie Modelu**: Model drzewa decyzyjnego został wytrenowany na zbiorze treningowym za pomocą biblioteki `scikit-learn`.
4. **Ewaluacja Modelu**: Model został przetestowany na zbiorze testowym w celu oceny jego skuteczności, mierzonej za pomocą metryk takich jak dokładność (accuracy).

4.1 Wyniki i Analiza

Wyniki klasyfikacji za pomocą drzewa decyzyjnego wskazują na umiarkowaną skuteczność tej metody. Dokładność klasyfikacji osiągnięta przez model wynosiła około 49%. Pomimo relatywnie prostego charakteru drzewa decyzyjnego, metoda ta może być podatna na przeuczenie w przypadku złożonych i różnorodnych danych tekstowych.



Rysunek 1: Graficzna reprezentacja drzewa decyzyjnego

5 Regresja logistyczna

Regresja logistyczna modeluje prawdopodobieństwo przynależności do określonej klasy jako funkcję logistyczną, która ma postać:

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n)}}$$

gdzie:

- $P(y = 1|X)$ to prawdopodobieństwo, że zmienna y przyjmuje wartość 1 (np. wypowiedź jest prawdziwa) dla danego zestawu cech X,
- $\beta_0, \beta_1, ..., \beta_n$ to współczynniki modelu,
- $X_1, X_2, ..., \beta_n$ to wartości cech dla danego przykładu.

5.1 Proces Budowy Modelu

1. **Przygotowanie Danych**: Podobnie jak w przypadku drzewa decyzyjnego, dane zostały wstępnie przetworzone, obejmując czyszczenie tekstu, tokenizację i konwersję do reprezentacji liczbowej za pomocą techniki TF-IDF.

2. **Podział Danych**: Zbiór danych został podzielony na zestawy treningowe i testowe.

3. **Trenowanie Modelu**: Model regresji logistycznej został wytrenowany na zbiorze treningowym za pomocą biblioteki `scikit-learn`.

4. **Ewaluacja Modelu**: Model został przetestowany na zbiorze testowym w celu oceny jego skuteczności.

5.2 Implementacja

Poniżej przedstawiono kroki implementacji regresji logistycznej w projekcie:

```
1. Przygotowanie Danych:
vectorization = TfidfVectorizer()
x_train_fit = vectorization.fit_transform(x_train)
x_test_fit = vectorization.transform(x_test)

2. Trenowanie Modelu:
logisticRegressionModel = LogisticRegression()
logisticRegressionModel.fit(x_train_fit, y_train)

3. Ewaluacja Modelu:
train_accuracy = accuracy_score(y_train, logisticRegressionModel.predict(x_train_fit))
test_accuracy = accuracy_score(y_test, logisticRegressionModel.predict(x_test_fit))

print(f"Accuracy for training data: {train_accuracy}")
print(f"Accuracy for testing data: {test_accuracy}")
```

5.3 Wyniki i Analiza

Wyniki klasyfikacji za pomocą regresji logistycznej wskazują na wyższą skuteczność tej metody w porównaniu do drzewa decyzyjnego. Model regresji logistycznej osiągnął dokładność około 63% na zbiorze testowym, co jest istotnym poprawieniem w stosunku do drzewa decyzyjnego.

5.4 Zalety i Ograniczenia

Regresja logistyczna ma kilka zalet:

- **Prostota i interpretowalność**: Model jest łatwy do interpretacji, a współczynniki można bezpośrednio interpretować jako wpływ poszczególnych cech na prawdopodobieństwo klasyfikacji.
- **Skalowalność**: Model dobrze działa na dużych zbiorach danych i może być stosunkowo szybko trenowany.

Ograniczenia regresji logistycznej obejmują:

- **Liniowość**: Model zakłada liniową zależność między cechami a logitami, co może być ograniczeniem w przypadku bardziej złożonych zależności.
- **Wrażliwość na niezrównoważone dane**: Model może mieć trudności z klasyfikacją w przypadku mocno niezrównoważonych zbiorów danych.

6 LSTM

Sieć LSTM składa się z jednostek (komórek) pamięci, które są odpowiedzialne za przechowywanie i manipulację stanem wewnętrznym. Każda jednostka LSTM zawiera trzy główne bramki, które kontrolują przepływ informacji:

- **Bramka wejściowa (input gate)**: Decyduje, które nowe informacje zostaną dodane do stanu komórki.
- **Bramka zapominania (forget gate)**: Decyduje, które informacje w stanie komórki zostaną zapomniane.
- **Bramka wyjściowa (output gate)**: Decyduje, które informacje z komórki zostaną wykorzystane do obliczenia wyjścia.

6.1 Struktura Sieci Bidirectional LSTM

Bidirectional LSTM składa się z dwóch warstw LSTM:

- **Forward LSTM**: Przetwarza sekwencje w standardowy sposób, od początku do końca.
- **Backward LSTM**: Przetwarza sekwencje w odwrotnym kierunku, od końca do początku.

Wyjścia z obu warstw są następnie łączone, co pozwala modelowi na dostęp do pełnego kontekstu sekwencji.

6.2 Proces Budowy Modelu LSTM i Bidirectional LSTM

1. **Przygotowanie Danych**: Dane tekstowe są przetwarzane, w tym czyszczenie tekstu, tokenizacja i konwersja do sekwencji liczbowych. Sekwencje te są następnie paddingowane, aby miały jednolitą długość.

2. **Tworzenie Modelu**: Model LSTM lub Bidirectional LSTM jest tworzony za pomocą frameworku Keras lub TensorFlow.

3. **Trenowanie Modelu**: Model jest trenowany na zbiorze treningowym przy użyciu optymalizatora (np. Adam) i funkcji straty (np. categorical cross-entropy).

4. **Ewaluacja Modelu**: Model jest testowany na zbiorze testowym w celu oceny jego skuteczności.

6.3 Implementacja

Poniżej przedstawiono kroki implementacji sieci LSTM i Bidirectional LSTM w projekcie "MIO Fake Detector":

```
1. Tworzenie Modelu LSTM:
sequentialModel = Sequential()
sequentialModel.add(Embedding(voc_size, embedding_vector_features,
input_length = sent_length))
sequentialModel.add(LSTM(100))
sequentialModel.add(Dense(4, activation = 'sigmoid'))
sequentialModel.compile(loss = 'binary_crossentropy',
optimizer = 'adam', metrics=['accuracy'])

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

2. Tworzenie Modelu Bidirectional LSTM:
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
sequentialModel1 = Sequential()
sequentialModel1.add(Embedding(voc_size,
embedding_vector_features, input_length=sent_length))
sequentialModel1.add(Bidirectional(LSTM(10)))
sequentialModel1.add(Dropout(0.3))
sequentialModel1.add(Dense(4, activation='softmax'))
sequentialModel1.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
print(sequentialModel1.summary())

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

3. Trenowanie Modelu LSTM:
sequentialModel.fit(X_train, Y_train, validation_data = (X_test, Y_test),
epochs = 10, batch_size=64)

4. Trenowanie Modelu Bidirectional LSTM
sequentialModel1.fit(X_train, Y_train, validation_data=(X_test, Y_test),
```

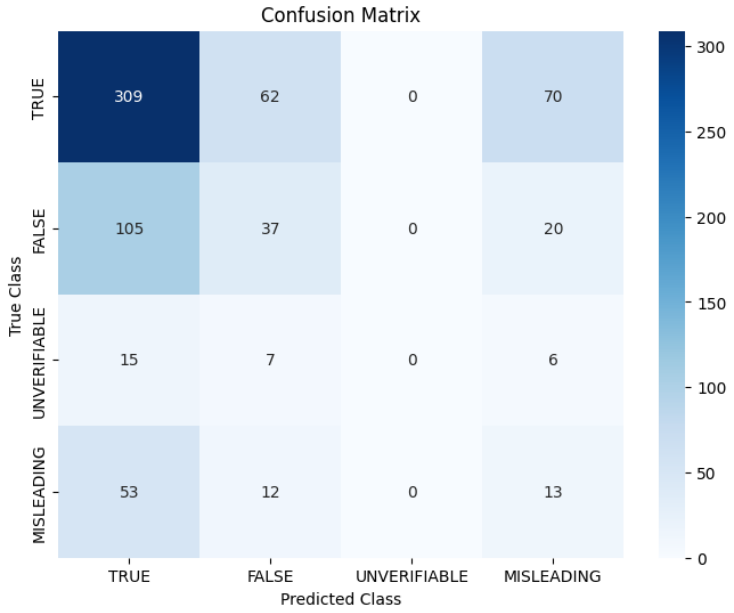
```
epochs=10, batch_size=64, callbacks=[early_stopping])

5. Ewaluacja Modelu:
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test.argmax(axis=1), y_pred.argmax(axis=1))
print(f"Accuracy: {accuracy}")
```

6.4 Wyniki i Analiza

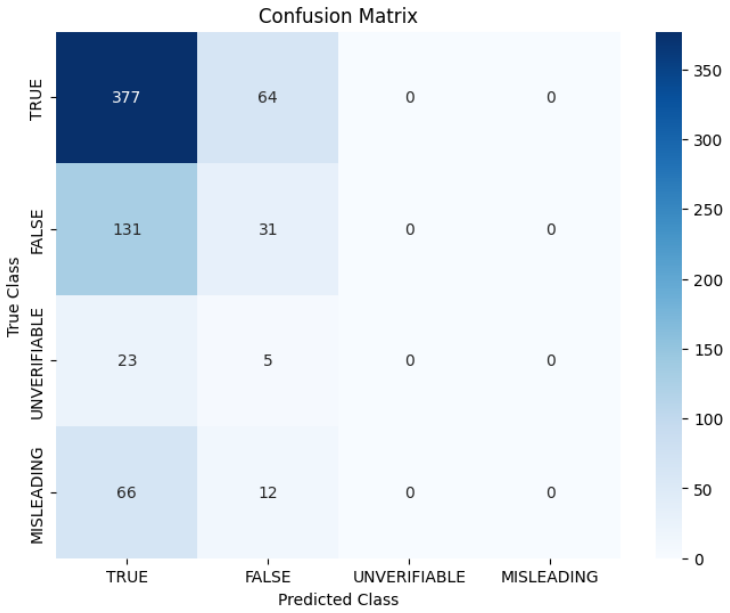
Wyniki klasyfikacji za pomocą sieci LSTM i Bidirectional LSTM w projekcie wskazyują na średnią skuteczność. Model opart o LSTM osiągnął wynik na poziomie 61%. Model oparty na Bidirectional osiągną wynik około 59%. Modele te są bardzo podatne na przetrenowanie.

6.5 Zalety i Ograniczenia

Mimo że LSTM osiągnął wyższą dokładność na zbiorze treningowym, problem przeuczenia wpłynął negatywnie na wyniki na zbiorze walidacyjnym.



Rysunek 2: Macierz pomyłek dla LSTM



Rysunek 3: Macierz pomyłek dla Bidirectional LSTM

7 Podsumowanie

Projekt miał na celu stworzenie skutecznego narzędzia do automatycznej klasyfikacji wypowiedzi politycznych pod kątem ich prawdziwości. W ramach projektu zastosowano kilka zaawansowanych technik z zakresu uczenia maszynowego i przetwarzania języka naturalnego (NLP).

- **Drzewa Decyzyjne**: Osiągnęły umiarkowaną skuteczność z dokładnością około 49%. Są łatwe do interpretacji, ale podatne na przeuczenie.
- **Regresja Logistyczna**: Uzyskała dokładność około 63%. Jest prosta i skalowalna, ale zakłada liniową zależność między cechami.
- **Sieci LSTM**: Osiągnęły dokładność około 61%.
- **Bidirectional LSTM**: Osiągnęły dokładność około 59%.

Zgodnie z publikacją "Discussion Manipulation, Language and Domain Dependent Models: An Overview"napisaną przez Filip Chalás, Igor Stupavský, Valentino Vranić można zgodzić się, że najlepsze wartości otrzymywane są na regresji liniowej. Modele przedstawione w publikacji były dla danych z datasetu dla języka słowackiego, który jest 6 razy większy. Przetestowaliśmy nawet przedstawione w publikacji modele, natomiast wszystko wskazuje na to, że dataset w języku polskim jest za mały.

Jako wniosek można uznać, że brakuje dobrego datasetu w języku polskim do trenowania takich danych. Wypowiedzi są za krótkie, aby otrzymać model z dobrą dokładnością i niezawodnością. Dane otrzymane przez Słowaków są delikatnie lepsze ze względu na to, że dane ich języku są większe niż w języku polskim.