Uniwersytet Rzeszowski

Wydział Nauk Ścisłych i Technicznych

Instytut Informatyki

Krzysztof Motas

Analiza wydźwięku tweetów na platformie X dotyczących transferów Realu Madryt

Projekt na ocenę 5 z przedmiotu Eksploracja danych internetowych

Prowadzący: dr inż. Piotr Lasek

Rzeszów 2025

Spis treści

1. Opis ontologii

Celem projektu było przeprowadzenie analizy wydźwięku tweetów użytkowników platformy X (dawniej Twitter), które odnoszą się do tematu transferów piłkarskich związanych z klubem Real Madryt. Głównym założeniem było stworzenie kompletnego procesu analitycznego: począwszy od zbierania danych, poprzez ich czyszczenie i analizę nastrojów, aż po wizualizację wyników oraz wyciągnięcie wniosków.

2. Proces zbierania danych

Dane zostały pozyskane z oficjalnego API platformy X, korzystając z wersji v2 i endpointu search/recent. W tym celu założono aplikację deweloperską na stronie developer.x.com oraz wygenerowano token autoryzacyjny, który umożliwia wykonywanie zapytań z poziomu aplikacji.

Aby pobrać posty dotyczące Realu Madryt i transferów, przygotowano specjalnie sformułowane zapytanie. Uwzględnia ono zarówno oficjalną nazwę klubu oraz jego hasztag, jak również popularne słowa i nazwiska związane z tematyką transferową. Zapytanie ograniczono do języka angielskiego, by ułatwić dalszą analizę nastrojów, i wykluczono retweety, które mogłyby zafałszować wyniki.

Użyte zapytanie wyglądało następująco:

```
search_url = "https://api.twitter.com/2/tweets/search/recent"

# Opcjonalne parametry: start_time,end_time,since_id,until_id,max_results,next_token,
# expansions,tweet.fields,media.fields,poll.fields,place.fields,user.fields
query_params = {
    'query': (
        '(#RealMadrid OR "Real Madrid") '
        '(transfer OR transfers OR sign OR signing OR signed OR deal OR bid OR '
        'rumor OR rumours OR target OR "Huijsen" OR "Trent Alexander-Arnold" OR "Carreras") '
        'lang:en -is:retweet'
    ),
    'max_results': '100',
    'tweet.fields': 'created_at,text,lang,author_id'
}
```

Rysunek 1. Zapytanie do API X

Parametry `tweet.fields` zapewniły, że dla każdego posta otrzymano informacje takie jak treść tweeta (`text`), czas jego opublikowania (`created_at`), język (`lang`) oraz identyfikator użytkownika (`author_id`).

Dane zostały następnie zapisane w pliku JSON i posłużyły jako plik wejściowy do dalszego przetwarzania. Ograniczenia darmowego poziomu API wymusiły przemyślane zapytanie i selekcję tematów.

Cały kod odpowiedzialny za komunikację z API znajduje się w pliku `fetch_tweets.py`.

3. Czyszczenie i przygotowanie danych

Przed przystąpieniem do analizy nastrojów, zebrane tweety zostały oczyszczone i przygotowane do przetwarzania. Głównym celem tego etapu było usunięcie zbędnych lub zakłócających elementów, takich jak odnośniki, znaki interpunkcyjne czy tzw. słowa puste (ang. *stop words*), które nie niosą istotnej wartości informacyjnej.

Operacje czyszczenia wykonano w pliku `clean_tweets.py` przy użyciu biblioteki `re` (do wyrażeń regularnych), string, nltk (do tokenizacji i usuwania słów pustych) oraz pandas. Proces obejmował następujące kroki:

- konwersję tekstu do małych liter,
- usunięcie adresów URL,
- usunięcie znaków interpunkcyjnych,
- tokenizację tekstu (podział na słowa),
- odfiltrowanie słów pustych z języka angielskiego.

Wynik oczyszczonego tekstu został zapisany w nowej kolumnie `cleaned_text`. Ostateczny wynik czyszczenia zapisano do pliku `tweets_cleaned.json`, który został użyty w dalszych etapach analizy.

Poniżej przedstawiono opisany fragment kodu:

```python
# Funkcja do wstępnego czyszczenia tekstu tweeta (małe litery, usunięcie URL-i i interpunkcji)
def clean_tweet(text):
    text = text.lower()
    text = re.sub( pattern: r'http\S+|www\S+|https\S+', repl: '', text, flags=re.MULTILINE)
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text

# Funkcja do usuwania słów bez znaczenia (stop words)
def remove_stopwords(text):
    tokens = word_tokenize(text)
    filtered = [word for word in tokens if word not in stop_words and word.isalpha()]
    return " ".join(filtered)

nltk.download('stopwords')
nltk.download('punkt')

with open("tweets.json", encoding="utf-8") as f:
    raw = json.load(f)

tweets_list = raw["data"]
df = pd.DataFrame(tweets_list)

df["cleaned_text"] = df["text"].apply(clean_tweet)
stop_words = set(stopwords.words("english"))
df["cleaned_text"] = df["cleaned_text"].apply(remove_stopwords)
```

Rysunek 2. Fragment pliku clean_tweets.py

4. Analiza wydźwięku tweetów

Analizę nastroju użytkowników przeprowadzono przy użyciu modelu VADER, który jest częścią biblioteki nltk.sentiment. VADER jest specjalnie przystosowany do pracy z krótkimi tekstami i językiem potocznym, dzięki czemu bardzo dobrze sprawdza się w analizie treści pochodzących z mediów społecznościowych.

Dla każdego tweeta obliczana była wartość `compound`, czyli wskaźnik nastroju mieszczący się w zakresie od -1 (bardzo negatywny) do +1 (bardzo pozytywny). Na jego podstawie przypisywano klasę nastroju:

- wynik ≥ 0.05 oznacza nastrój pozytywny,
- wynik ≤ -0.05 oznacza nastrój negatywny,
- wartości pośrednie interpretowano jako neutralne.

Analiza została zrealizowana w pliku `analyze_sentiment.py`, a wyniki zapisano w dwóch kolumnach: `sentiment` (wartość liczbowa) oraz `sentiment_label` (etykieta: `positive`, `neutral`, `negative`).

Opisywany fragment kodu został umieszczony poniżej:

```python
# Inicjalizacja VADER
sia = SentimentIntensityAnalyzer()

# Oblicza ogólny wynik nastroju dla podanego tekstu
def get_vader_sentiment(text):
    return sia.polarity_scores(text)["compound"]

# Klasyfikuje wynik nastroju jako pozytywny, neutralny lub negatywny
def classify_sentiment(score):
    if score >= 0.05:
        return "positive"
    elif score <= -0.05:
        return "negative"
    else:
        return "neutral"

df["sentiment"] = df["cleaned_text"].apply(get_vader_sentiment)
df["sentiment_label"] = df["sentiment"].apply(classify_sentiment)
```

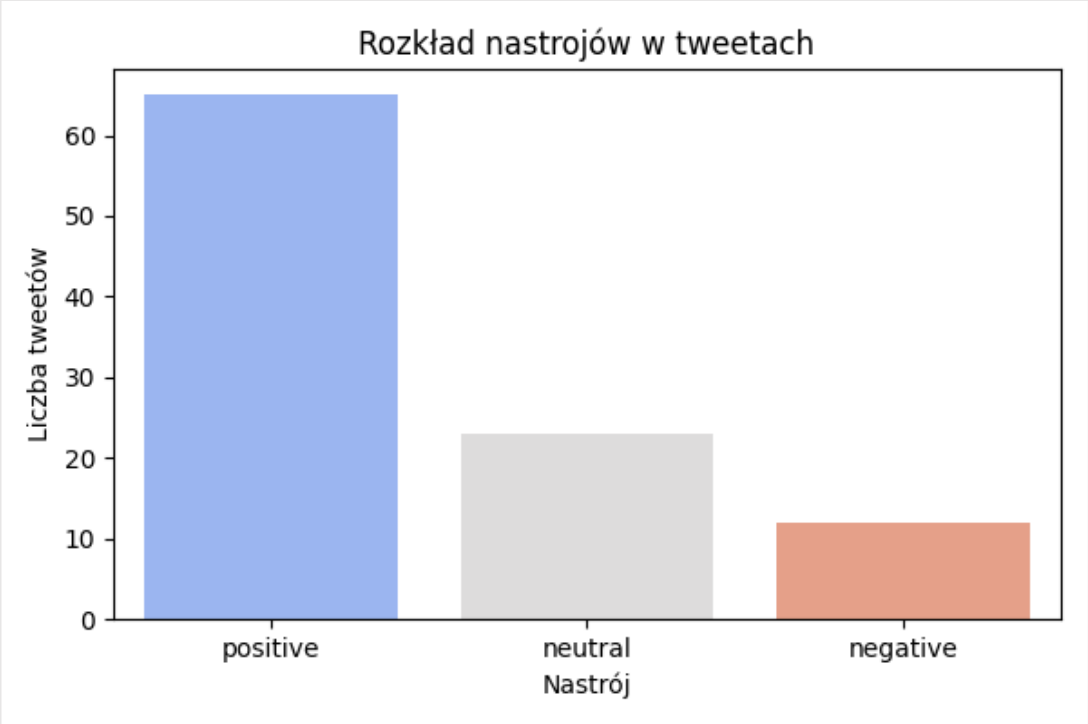
Rysunek 3. Fragment pliku analyze_sentiment.py

5. Wizualizacja wyników

W celu interpretacji wyników analizy przygotowano cztery wykresy z użyciem bibliotek `matplotlib`, `seaborn` oraz `wordcloud`. Każdy z nich przedstawia inne aspekty nastrojów użytkowników wobec transferów Realu Madryt.

- **Wykres słupkowy – Rozkład klas nastrojów**

Ten wykres przedstawia liczbę tweetów w każdej klasie nastroju: pozytywne, neutralne, negatywne. Pozwala on szybko określić, który typ emocji przeważał wśród użytkowników. W analizowanym zbiorze dominowały tweety pozytywne, co sugeruje optymistyczne nastawienie fanów klubu wobec możliwych transferów.



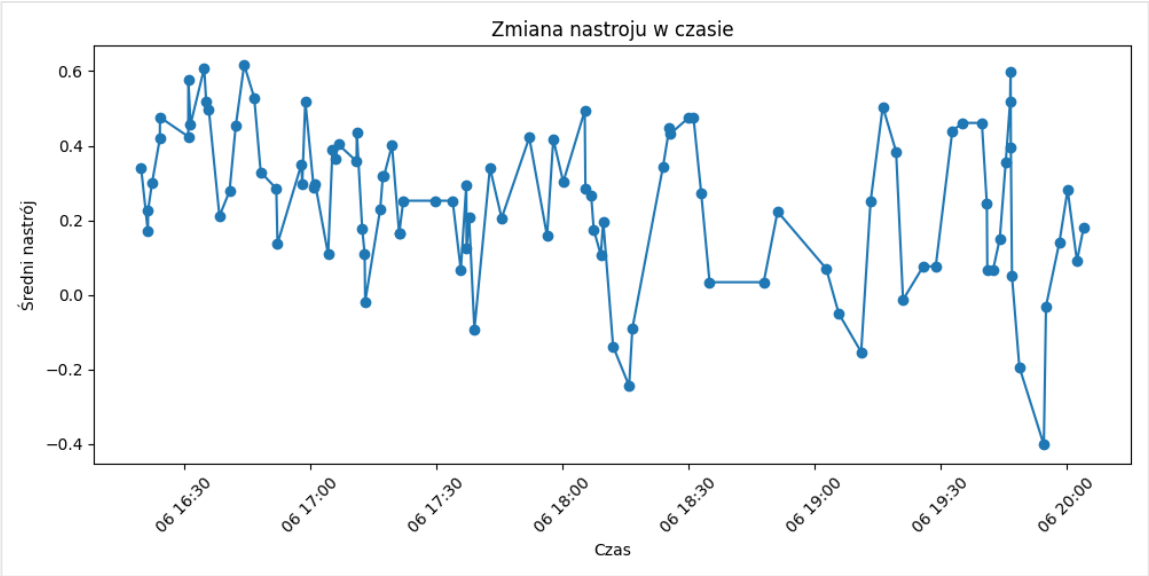
Rysunek 4. Wykres słupkowy przedstawiający rozkład klas nastrojów

```
# Wykres słupkowy - rozkład nastrojów
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x="sentiment_label", hue="sentiment_label", palette="coolwarm", legend=False)
plt.title("Rozkład nastrojów w tweetach")
plt.xlabel("Nastrój")
plt.ylabel("Liczba tweetów")
plt.tight_layout()
plt.show()
```

Rysunek 5. Kod wykresu słupkowego

- **Wykres liniowy – Średni nastrój w czasie**

Wykres ten przedstawia, jak zmieniała się wartość średniego nastroju w czasie. Dane zagregowano godzinowo. W przypadku badanej próbki, tweety pochodziły z jednego dnia, co ograniczyło zakres czasowy analizy, ale pozwoliło zaobserwować niewielkie zmiany emocji w ciągu dnia.



Rysunek 6. Wykres liniowy przedstawiający zmianę nastrojów w czasie

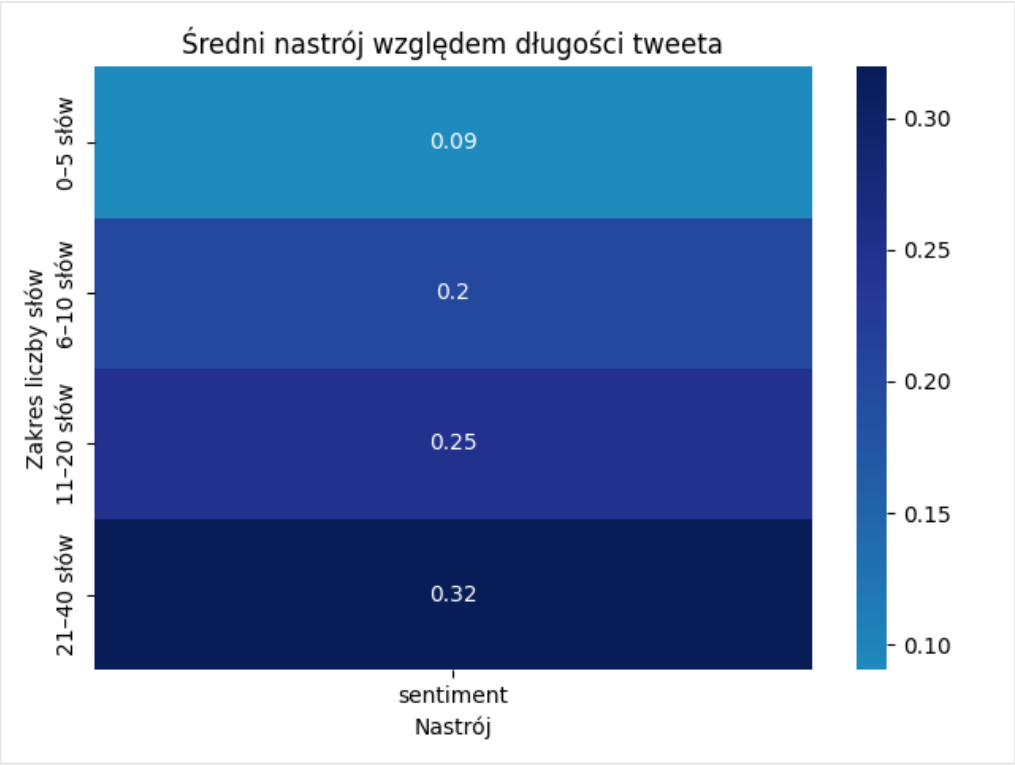
```
# Wykres liniowy - nastrój w czasie
df_sorted = df.sort_values("created_at")
df_sorted["avg_sentiment"] = df_sorted["sentiment"].rolling(window=3, min_periods=1).mean()

plt.figure(figsize=(10, 5))
plt.plot( *args: df_sorted["created_at"], df_sorted["avg_sentiment"], marker="o")
plt.title("Zmiana nastroju w czasie")
plt.xlabel("Czas")
plt.ylabel("Średni nastrój")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Rysunek 7. Kod wykresu liniowego

- **Heatmapa – Nastrój względem długości tweeta**

Mapa cieplna pokazuje, jak kształtował się średni nastrój w zależności od liczby słów w tweetach. Dane pogrupowano na cztery przedziały długości (np. 0–5, 6–10 słów itd.). Wykres ten pozwala dostrzec, czy bardziej rozbudowane wypowiedzi miały inny charakter emocjonalny niż krótkie. Można zauważyć, że dłuższe tweety często miały bardziej pozytywny wydźwięk.



Rysunek 8. Mapa cieplna przedstawiająca średni nastrój względem długości tweeta

```python
# Heat mapa - średni nastrój vs. długość tweeta
df["word_count"] = df["cleaned_text"].apply(lambda x: len(x.split()))

df["word_count_range"] = pd.cut(
    df["word_count"],
    bins=[0, 5, 10, 20, 40],
    labels=["0-5 słów", "ó-10 słów", "11-20 słów", "21-40 słów"]
)

avg_sentiment_by_length = df.pivot_table(
    index="word_count_range",
    values="sentiment",
    aggfunc="mean",
    observed=False
)

sns.heatmap(avg_sentiment_by_length, annot=True, cmap="YlGnBu", center=0)
plt.title("Średni nastrój względem długości tweeta")
plt.ylabel("Zakres liczby słów")
plt.xlabel("Nastrój")
plt.tight_layout()
plt.show()
```

Rysunek 9. Kod heatmapy

- **Chmura słów – Najczęściej używane słowa**

Chmura słów pokazuje, które słowa pojawiały się najczęściej w tweetach po czyszczeniu. Jest to przydatne narzędzie eksploracyjne, pozwalające zidentyfikować główne nazwiska, hasła

i tematy poruszane przez użytkowników. W chmurze dominowały wyrazy takie jak "real", "madrid", "window", a także nazwiska konkretnych piłkarzy.

Rysunek 10. Chmura słów

```
# Chmura słów
text = " ".join(df["cleaned_text"])
wordcloud = WordCloud(width=800, height=400, background_color="white", colormap="Blues").generate(text)
```

Rysunek 11. Kod chmury słów

6. Wnioski

Zrealizowany projekt pokazał, że analiza nastrojów na podstawie danych z platformy X może być przydatnym narzędziem do oceny ogólnego tonu wypowiedzi użytkowników. Pomimo ograniczeń w liczbie dostępnych danych, możliwe było zaobserwowanie podstawowych tendencji emocjonalnych. Projekt potwierdził także skuteczność prostych metod przetwarzania języka naturalnego i wizualizacji w pracy z treściami z mediów społecznościowych.