

# **Analiza polskich tekstów z Twittera**

Kamil Chlebek  
Szymon Nowak

# 1. Źródło danych

## Opis sposobu zbierania danych

Dane do analizy zostały pobrane za pomocą biblioteki TwitterSearch zaimplementowanej w języku python. Biblioteka ta, komunikuje się z Twitter API umożliwiając pobieranie tweetów spełniających różne kryteria (występujące słowa, tagi itd).

### Instrukcja

1. Aby móc korzystać z Twitter API, należy zarejestrować się na <https://apps.twitter.com/> oraz utworzyć aplikację. Następnie należy pobrać "consumer\_key", "consumer\_secret", "access\_token" oraz "access\_token\_secret". Wymienione dane pozwolą na komunikację z Twitter API.
2. Instalacja biblioteki TwitterSearch za pomocą pip install TwitterSearch
3. Utworzenie kryteriów wyszukiwania tweetów zgodnie z opisem w dokumentacji, tj. utworzenie obiektu TwitterSearchOrder
4. Utworzenie obiektu TwitterSearch, który komunikuje się z Twitter API.
5. Pobranie i zapisanie danych.

Aby pobrać próbkę tweetów dla danego użytkownika należy uruchomić python3.5 tweet\_download\_user.py <<nazwa użytkownika>> np: python3.5 tweet\_download\_user.py JaroslawKuzniar, tweety zostaną zapisane w pliku tweets\_<<autor>>.csv

Aby pobrać próbkę tweetów różnych autorów należy uruchomić python3.5 tweet\_download.py, tweety zostaną zapisane w pliku tweets.csv

## 2. Analiza stylu

### 1. Cel

Celem tej części projektu jest hierarchiczne grupowanie użytkowników według podobieństwa cech stylistycznych oraz zbudowanie modelu, który dla danego tekstu wejściowego jest w stanie przyporządkować odpowiadającego mu autora biorąc pod uwagę styl wypowiedzi.

### 2. Dane wejściowe

W celu przeprowadzenia eksperymentu pobrano po 700 tweetów dla użytkowników BoniekZibi, JaroslawKuzniar, robertbiedron, sikorskiradek. Pobrane dane zostały zapisane w pliku csv w formacie: autor, tekst, data, ilość retweetów, ilość polubień, ilość followersów autora, lista tagów. np.: "BoniekZibi"; "Historia wszystkich meczów Reprezentacji, polecam. Świeży prezent!  
<https://t.co/0rOK7h3ONd>"; "Sun Dec 03 08:55:16 +0000 2017"; "13"; "92"; "929415"; ""

Dla każdego z autorów, zbiór tweetów został podzielony na dwa zbiory po 350 tweetów każdy. Jeden który posłuży do utworzenia modelu i drugi w celu przeprowadzenia testów.

### 3. Koncepcja rozwiązania

Koncepcja rozwiązania, zakłada wygenerowanie dla każdego autora na podstawie jego wypowiedzi wektora, który reprezentuje jego styl. Następnie posiadając takie wektory, można zgrupować autorów o podobnym stylu, lub przewidzieć jaki jest autor podając jedynie tekst wypowiedzi.

### 4. Przygotowanie danych

Wstępne przetworzenie danych polega na usunięciu z tekstu tweeta wszystkich linków, tagów, oznaczeń innych użytkowników, znaków interpunkcyjnych oraz emoji.

### 5. Zastosowane rozwiązania

W celu wygenerowania wektora użyto ParagraphVectors z deeplearning4j, która działa podobnie jak Word2Vec, z tym że zamiast określać wzajemne relacje między słowami, określa relacje między dokumentami (w naszym przypadku wypowiedziami na tweeterze) do których przypisane są etykiety (autory). W celu wyznaczenia wektora, deeplearning4j domyślnie dzieli wypowiedzi na pojedyncze słowa (DefaultTokenizer), oprócz domyślnego tokenizera wykorzystano NGramTokenizer, który dzieli wypowiedzi na n-gramy.

Do pobrania części mowy dla danego słowa użyto biblioteki morfologic.

Klasteryzacja hierarchiczna została wykonana z użyciem <https://github.com/lbehnke/hierarchical-clustering-java>, który wychodząc od pojedynczego wektora łączy podobne wektory ze sobą.

## 6. Sposób oceny rezultatów

Po utworzeniu modelu na podstawie 50% pobranych tweetów, dla pozostałych 50% tweetów wyznaczono najbardziej prawdopodobnego autora. Ocena rezultatu (0-1) to stosunek liczby tweetów dla którego przewidziany autor zgadzał się z autorem faktycznym, do liczby wszystkich przetwarzanych tweetów ze zbioru testowego.

## 7. Wyniki eksperymentów

VectorConfig{minWordFrequency=1, iterations=1, epochs=5, layerSize=100, learningRate=0.25, windowSize=5, trainWordVectors=false, sampling=0}

Nazwa eksperymentu	Prawidłowe	Nieprawidłowe	Ocena
Tekst - DefaultTokenizer	704	696	0.5028571428571429
Tekst - NGramTokenizer	619	781	0.4421428571428571
POS - DefaultTokenizer	508	892	0.3628571428571429
POS - NGramTokenizer	474	926	0.3385714285714286

VectorConfig{minWordFrequency=1, iterations=1, epochs=4, layerSize=100, learningRate=0.12, windowSize=5, trainWordVectors=false, sampling=0}

Nazwa eksperymentu	Prawidłowe	Nieprawidłowe	Ocena
Tekst - DefaultTokenizer	731	669	0.5221428571428571
Tekst - NGramTokenizer	664	736	0.4742857142857143
POS - DefaultTokenizer	526	874	0.3757142857142857
POS - NGramTokenizer	504	896	0.36

VectorConfig{minWordFrequency=1, iterations=5, epochs=5, layerSize=100, learningRate=0.25, windowSize=3, trainWordVectors=false, sampling=0}

Nazwa eksperymentu	Prawidłowe	Nieprawidłowe	Ocena
Tekst - DefaultTokenizer	644	756	0.46
Tekst - NGramTokenizer	647	753	0.46214285714285713
POS - DefaultTokenizer	540	860	0.38571428571428573
POS - NGramTokenizer	479	921	0.34214285714285714

VectorConfig{minWordFrequency=1, iterations=1, epochs=7, layerSize=500, learningRate=0.1, windowSize=5, trainWordVectors=false, sampling=0}

Nazwa eksperymentu	Prawidłowe	Nieprawidłowe	Ocena
Tekst - DefaultTokenizer	754	646	0.5385714285714286
<b>Tekst - NGramTokenizer</b>	<b>643</b>	<b>757</b>	<b>0.5407142857142857</b>
<b>POS - DefaultTokenizer</b>	<b>549</b>	<b>851</b>	<b>0.3921428571428571</b>
POS - NGramTokenizer	512	888	0.3657142857142857

Tekst - NGramTokenizer (najlepsza ocena dla samego tekstu)



POS - DefaultTokenizer (najlepsza ocena dla części mowy)



## 3. Analiza sentymenu

### 1. Krótki opis

Celem tej części projektu jest wyznaczenie sentymenu dla zbioru tweetów na podstawie wcześniej przygotowanego zbioru uczącego, a następnie ręczna weryfikacja skuteczności. Określanie sentymenu polegało na przypisaniu jednej z trzech etykiet:

- POS - sentyment pozytywny
- NEU - sentyment neutralny
- NEG - sentyment negatywny

Danymi wejściowymi były tweety użytkowników BoniekZibi, JaroslawKuzniar, robertbiedron, sikorskiradek - oba zbiory (testowy i uczący) zostały wybrane z tego zbioru.

Zadanie zostało rozbite na następujące części:

- Przygotowanie zbioru uczącego
- Przygotowanie wektora
- Weryfikacja wektora na zbiorze uczącym
- Weryfikacja wektora na zbiorze testowym

### 2. Przygotowanie zbioru uczącego

Ze zbioru wszystkich tweetów wybrane zostało 150 tweetów, które pełniło rolę zbioru uczącego. Tweety zostały wybrane tak, aby występowała tam taka sama ilość tweetów dla danej etykiety sentymenu (po 50 tweetów dla POS, NEU i NEG). Wybrane tweety zostały zapisane w pliku "sentiment\_train\_data.csv".

Następnie zbiór uczący został załadowany do programu i poddany obróbce w celu usunięcia słów/znaków, które zmniejszałyby skuteczność etykietowania - z tweetów zostały usunięte linki, słowa zaczynające się od @ (odnośnik do innego użytkownika) i # (hashtag). Wszystkie znaki zostały zamienione na małe, dodatkowo ze słów zostały usunięte wszystkie znaki, które nie są literami - dzięki temu udało się ze zbioru uczącego usunąć znaki interpunkcyjne i emoji. Na koniec wszystkie wielokrotne znaki spacji zostały zamienione na pojedyńczy znak.

Następnie, w celu zwiększenia skuteczności algorytmu etykietującego, słowa z tweetów zostały dodatkowo przetworzone przy użyciu słownika biblioteki morfologik. Dla każdego słowa zostały pobrane jego formy podstawowe (dla przykładu słowo "mnie" ma dwie formy podstawowe - "ja" i "mięć"), posortowane alfabetycznie i połączone ze sobą. Następnie słowo z tweeta zostało zastępowane przez jego formy podstawowe.

#### Przykładowe przetworzenie:

Wersja podstawowa:

"KOGUT pieje z radości ! <https://t.co/0aSRL7Zm8k> 😂😂😂"

Wersja po przetworzeniu:

“*kogut piać z radość*”

Wersja podstawowa:

“*Widziałem naprawdę młodego polskiego piłkarza z wielką smykałką -KK kiedy? Gdzie?  
Nic nie powiem!*”

Wersja przetworzona:

“*widzieć naprawdę młodemu polskiemu piłkarz z wielki smykałka kk kiedy gdzie nicnica  
nieon powiedzieć*”

### 3. Przygotowanie wektora

Przy użyciu przygotowanego zbioru uczącego wygenerowany został obiekt ParagraphVector z biblioteki DeepLearning4J (deeplearning4j.models.paragraphvectors.ParagraphVectors). Wektor został zbudowany przy użyciu ParagraphVectors.Builder, dostarczając mu dodatkowe informacje:

- obiekt typu Sentencelerator, zainicjalizowany przy użyciu zbioru tweetów ze zbioru uczącego
- obiekt typu LabelsSource, zainicjalizowany zbiorem etykiet
- obiekt przetwarzający tokeny o typie DefaultTokenizerFactory

W taki sposób skonfigurowany wektor został następnie zbudowany (przy użyciu metody .build()), a następnie “dopasowany” (przy użyciu metody .fit()).

### 4. Weryfikacja wektora na zbiorze uczącym

Weryfikacja zakładała wygenerowanie wektora w sposób opisany w poprzednim podpunkcie, a następnie wyznaczenie przy jego użyciu etykiety dla tweetów ze zbioru uczącego. Następnie zostało obliczane, z jaką skutecznością wektor wyznaczył etykiety dla zbioru uczącego. Do wyznaczania etykiety dla tweetów została wykorzystana metoda klasy ParagraphVector .nearestLabels(java.lang.String rawText, int topN).

Dla zbioru uczącego, przy podanej wyżej konfiguracji, skuteczność etykietowania obejmowała od 33% do 44%. Był to wynik wysoce niezadowalający, ponieważ przy 3 możliwych etykietach, szansa skutecznego wyznaczenia etykiety przy użyciu wektora była niedużo wyższa od losowego przydzielania etykiety. Postanowiono więc zmienić konfigurację generowanego wektora, żeby zwiększyć dokładność etykietowania.

Ostatecznie, ParagraphVector został skonfigurowany w następujący sposób:

- ilość iteracji dla każdego zbioru danych podczas uczenia - 1
- minimalna ilość występowania słowa - 1
- ilość iteracji po całym zbiorze uczącym (epochs) - 5
- ilość “wymiarów” wektora wyjściowego - 100
- współczynnik uczenia dla modelu trenującego - 0.4

- reprezentacje słów nie powinny być budowane razem z reprezentacjami dokumentu (`trainWordVectors = false`)
- wielkość okna kontekstu - 5

Przy wykorzystaniu takiej konfiguracji skuteczność etykietowania zbioru uczącego zwiększyła się do 88%

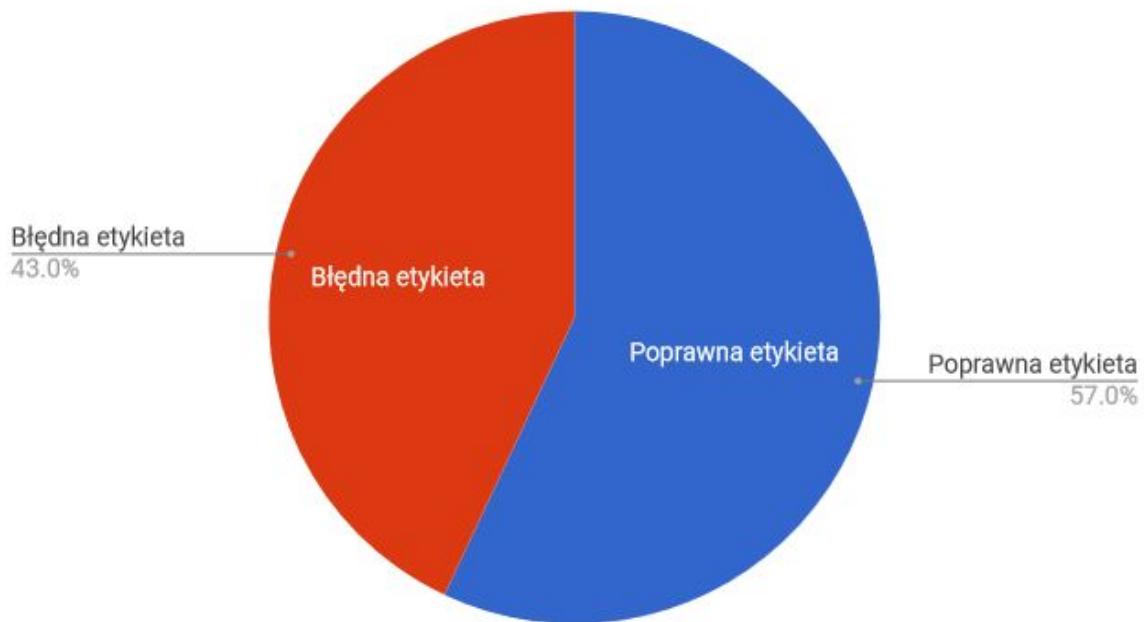
Czemu nie 100%? Może się to wiązać z tym, że jedno słowo może być wykorzystane w tweetach, dla których sentyment będzie różny. Dla przykładu słowo "pięknie" może zostać użyte w zdaniu "Pięknie dziś wyglądasz" (komplement, pozytywny) oraz w zdaniu "No to pięknie" (niemiła niespodzianka, wypadek, negatywne).

## 5. Weryfikacja wektora na zbiorze testowym

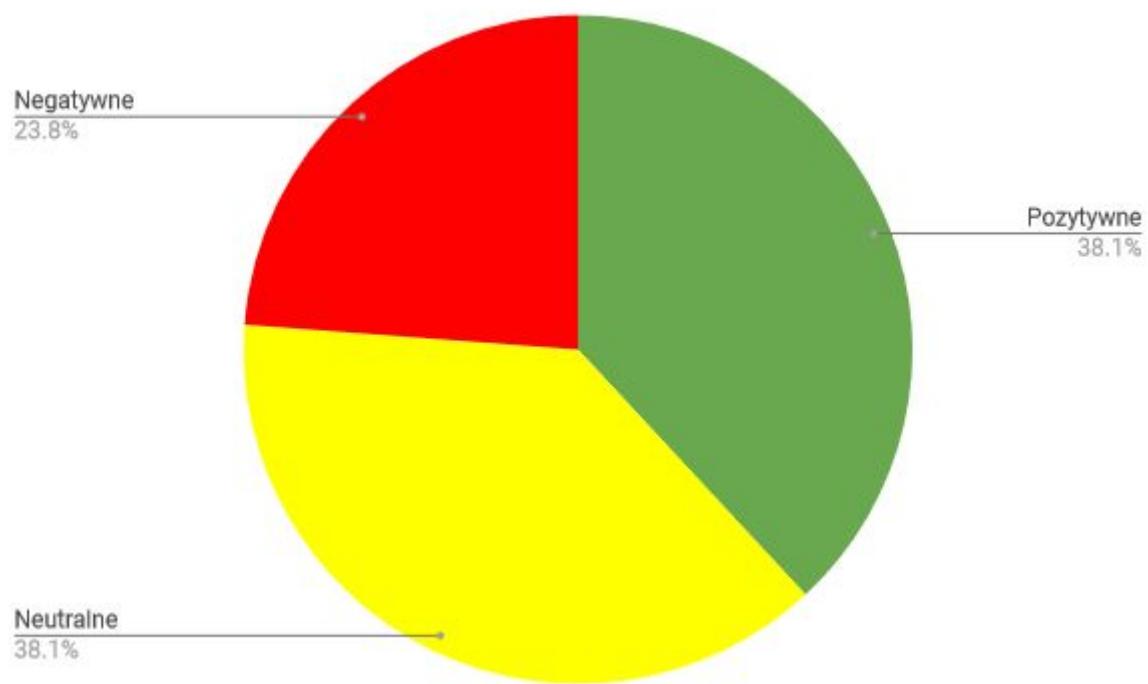
W celu przetestowania wygenerowanego wektoru, przygotowany został zbiór tweetów, w których nie znalazł się ani jeden tweet użyty w zbiorze tweetów uczących. Tweety zostały poddane tej samej obróbce, aby swoją formą nie odbiegały od modelu uczącego.

Wyniki eksperymentu zaprezentowane są na poniższym wykresach:

### Poprawność etykietowania:



### **Sentyment poprawnie etykietowanego tweeta:**



Analizując wyniki sentymentów poprawnie etykietowanego tweetów można wywnioskować, że nie ma mocnych przesłanek aby stwierdzić, że tweety o wybranym sentymencie są łatwiejsze do etykietowania przez wektor.