Sprawozdanie z Projektu: Przetwarzanie Języka Naturalnego

Autorzy: Jarosław Klima, Paweł Knot
Data: Luty 2026
Przedmiot: Przetwarzanie Języka Naturalnego (PJN)

Spis treści

---

1. Wprowadzenie

Niniejsze sprawozdanie dokumentuje kompleksowy projekt z zakresu przetwarzania języka naturalnego (ang. *Natural Language Processing*, NLP), zrealizowany w ramach przedmiotu PJN. Projekt składa się z trzech niezależnych, ale tematycznie powiązanych modułów, które przedstawiają różne aspekty analizy i generowania języka naturalnego.

Celem pracy było praktyczne zastosowanie metod i algorytmów NLP do rozwiązania trzech różnych problemów: analizy statystycznej korpusu językowego, interaktywnego generowania poprawnych gramatycznie zdań oraz analizy relacji semantycznych między częściami mowy. Każdy z podprojektów stanowi odrębną całość z własnymi celami, metodami i wynikami.

Projekt został zrealizowany w języku Python z wykorzystaniem nowoczesnych bibliotek i framework'ów specjalizujących się w przetwarzaniu języka naturalnego.

---

2. Cel projektu

Głównym celem projektu było:

1. **Zgłębienie praktycznych aspektów przetwarzania języka naturalnego** poprzez implementację różnorodnych algorytmów analizy i generowania tekstu.

2. **Zastosowanie metod statystycznych** do analizy korpusu językowego, w tym weryfikacja empiryczna prawa Zipfa oraz analiza częstotliwości występowania słów.
3. **Stworzenie interaktywnego narzędzia edukacyjnego** do nauki struktury gramatycznej języka angielskiego, bazującego na regułach gramatyki generatywnej.
4. **Przeprowadzenie analizy semantycznej** relacji między różnymi częściami mowy na podstawie dużego korpusu tekstowego.
5. **Praktyczne opanowanie narzędzi NLP** takich jak NLTK, spaCy, czy technik wizualizacji danych językowych.

---

3. Podział na podprojekty

Realizacja projektu została podzielona na trzy niezależne moduły:

Projekt 1: Analiza Korpusu Językowego

Typ: Analiza statystyczna
Zakres: Kompleksowa analiza korpusu 100,000 słów
Główne zadania:

- Analiza częstotliwości występowania wyrazów
- Weryfikacja prawa Zipfa
- Identyfikacja minimalnego zestawu słów pokrywających 90% tekstu
- Wizualizacja relacji między sąsiednimi słowami (graf sąsiedztwa)
- Ranking najczęstszych rzeczowników

Projekt 2: Generator Zdań SVO

Typ: Generowanie języka naturalnego
Zakres: Interaktywna aplikacja webowa do budowania zdań
Główne zadania:

- Implementacja struktury SVO (Subject-Verb-Object)
- Obsługa odmiany czasowników przez osoby i czasy
- Generowanie pytań i przeczeń
- Automatyczny dobór przedimków i form liczby mnogiej
- Stworzenie intuicyjnego interfejsu użytkownika

Projekt 3: Analiza Semantyczna

Typ: Analiza relacji semantycznych
Zakres: Badanie współwystępowania części mowy
Główne zadania:

- Identyfikacja części mowy (POS tagging)
- Tworzenie grafów dwudzielnych dla relacji przymiotnik-rzeczownik i czasownik-rzeczownik
- Wizualizacja macierzy współwystępowania
- Analiza kolokacji i związków frazeologicznych

4. Wymagania systemowe

Wymagania ogólne

- **System operacyjny:** Windows 10/11, Linux (Ubuntu 20.04+), macOS 11+
- **Python:** wersja 3.8 lub nowsza
- **RAM:** minimum 4 GB (zalecane 8 GB)
- **Miejsce na dysku:** około 500 MB (bez środowisk wirtualnych)
- **Przeglądarka:** Chrome, Firefox, Edge lub Safari (dla Projektu 2)

Biblioteki Python

Projekt 1:

```
matplotlib >= 3.5.0
pandas >= 1.5.0
numpy >= 1.23.0
nltk >= 3.8.0
networkx >= 3.0
```

Projekt 2:

```
streamlit
inflect
lemminflect
```

Projekt 3:

```
spacy >= 3.7.0
matplotlib >= 3.5.0
pandas >= 1.5.0
numpy >= 1.23.0
networkx >= 3.0
```

Dodatkowe zasoby:

- Model spaCy dla języka angielskiego: `en_core_web_sm`
- Korpus NLTK (pobierany automatycznie przy pierwszym uruchomieniu)

5. Instrukcja instalacji i uruchomienia

5.1 Przygotowanie środowiska

Każdy podprojekt powinien być uruchamiany w osobnym środowisku wirtualnym, aby uniknąć konfliktów zależności.

Krok 1: Utworzenie środowiska wirtualnego

W katalogu wybranego projektu:

```
python -m venv venv
```

Krok 2: Aktywacja środowiska

Windows:

```
venv\Scripts\activate
```

Linux/Mac:

```
source venv/bin/activate
```

5.2 Uruchomienie Projektu 1 - Analiza Korpusu

```
cd projekt_1
python -m venv venv
venv\Scripts\activate          # Windows
pip install -r requirements.txt
python analiza_korpusu.py
```

Wyniki: Program generuje pliki CSV oraz wykresy PNG w katalogu projektu.

5.3 Uruchomienie Projektu 2 - Generator Zdań SVO

```
cd projekt_2
python -m venv venv
venv\Scripts\activate          # Windows
pip install -r requirements.txt
streamlit run app.py
```

Interfejs: Aplikacja automatycznie otworzy się w przeglądarce pod adresem `http://localhost:8501`

5.4 Uruchomienie Projektu 3 - Analiza Semantyczna

```
cd projekt_3
python -m venv venv
venv\Scripts\activate          # Windows
pip install -r requirements.txt
python -m spacy download en_core_web_sm
python analiza_semantyczna.py
```

Wyniki: Program generuje grafy, macierze heatmap oraz pliki tekstowe z listami połączeń.

---

6. Opis realizacji i analiza wyników

6.1 Projekt 1 - Analiza Korpusu Językowego

6.1.1 Metodologia

Projekt wykorzystuje korpus składający się z 100,000 słów w języku angielskim. Proces analizy obejmuje:

1. **Preprocessing tekstu:**
    - Tokenizacja przy użyciu NLTK
    - Normalizacja (lowercase)
    - Usuwanie znaków interpunkcyjnych
2. **Analiza częstotliwości:**
    - Zliczanie wystąpień każdego słowa
    - Przypisanie rang (r) według częstotliwości
    - Obliczenie iloczynu r × f (test prawa Zipfa)
3. **Tagowanie części mowy (POS):**
    - Identyfikacja rzeczowników przy użyciu NLTK POS tagger
    - Ranking 50 najczęstszych rzeczowników
4. **Analiza grafowa:**
    - Budowa grafu skierowanego reprezentującego sąsiedztwo słów
    - Wizualizacja przy użyciu NetworkX

6.1.2 Wyniki

Prawo Zipfa:

Program generuje wykres log-log przedstawiający relację między rangą słowa a jego częstotliwością. Zgodnie z prawem Zipfa, w naturalnym języku częstotliwość słowa jest odwrotnie proporcjonalna do jego rangi. Wyniki z korpusu potwierdzają tę zależność, pokazując charakterystyczną linię prostą na wykresie logarytmicznym.

Analiza odcięcia 90%:

Program identyfikuje minimalny zestaw najczęstszych słów, które łącznie stanowią 90% całego korpusu. Ten wynik ilustruje zasadę Pareto w języku naturalnym - stosunkowo niewielka liczba słów (zazwyczaj około 2000-3000) pokrywa większość rzeczywistego użycia językowego.

Graf sąsiedztwa:

Wizualizacja pokazuje najczęstsze bigramy (pary sąsiednich słów), co pozwala zidentyfikować kolokacje i typowe związki frazeologiczne występujące w korpusie.

Top 50 rzeczowników:

Lista najczęstszych rzeczowników dostarcza informacji o tematyce korpusu oraz najważniejszych pojęciach w analizowanym tekście.

6.1.3 Wygenerowane pliki

- `analiza_czestotliwosci.csv` - kompletna tabela z rangami, częstotliwościami i iloczynami
- `wykres_zipfa.png` - wizualizacja prawa Zipfa
- `graf_sasiedztwa.png` - graf najczęstszych bigramów
- Wydruk top 50 rzeczowników w konsoli

6.2 Projekt 2 - Generator Zdań SVO

6.2.1 Architektura systemu

Aplikacja oparta na framework'u Streamlit implementuje gramatykę generatywną dla języka angielskiego. System składa się z trzech głównych modułów:

1. **Moduł danych (data.json):**
    - Baza 100 rzeczowników
    - Baza 100 czasowników
    - Baza 100 przymiotników
2. **Silnik gramatyczny:**
    - Odmiana czasowników przez osoby (1., 2., 3. osoba)
    - Obsługa trzech czasów: Present Simple, Past Simple, Future Simple
    - Obsługa czasowników nieregularnych w Past Simple
    - Generowanie pytań z inwersją i operatorami pomocniczymi
    - Generowanie przeczeń z odpowiednimi formami *not*
3. **Moduł morfologiczny:**
    - Automatyczny dobór przedimków *a/an* (biblioteka inflect)
    - Tworzenie liczby mnogiej rzeczowników
    - Odmiana czasowników (biblioteka lemminflect)

6.2.2 Interfejs użytkownika

Aplikacja prowadzi użytkownika przez proces budowania zdania w trzech krokach:

Krok 1: Wybór podmiotu

- Wybór rzeczownika z listy rozwijanej
- Opcja dodania przymiotnika
- Wybór liczby (pojedyncza/mnoga)
- Automatyczny dobór przedimka

Krok 2: Wybór czasownika

- Wybór czasownika z listy
- Wybór czasu (Present/Past/Future)
- Wybór typu zdania (twierdzące/pytające/przeczące)

Krok 3: Wybór dopełnienia

- Wybór rzeczownika
- Opcja przymiotnika
- Wybór liczby
- Automatyczny dobór przedimka

Prezentacja wyniku: Aplikacja wyświetla ostateczne zdanie z automatycznie zastosowanymi wszystkimi regułami gramatycznymi.

6.2.3 Przykłady generowanych zdań

```
Twierdzenie Present: "A beautiful cat eats an apple."
Pytanie Present: "Does a beautiful cat eat an apple?"
Przeczenie Present: "A beautiful cat does not eat an apple."

Twierdzenie Past: "The children saw a movie."
Pytanie Past: "Did the children see a movie?"
Przeczenie Past: "The children did not see a movie."

Twierdzenie Future: "The teacher will read books."
Pytanie Future: "Will the teacher read books?"
Przeczenie Future: "The teacher will not read books."
```

6.2.4 Znaczenie edukacyjne

Aplikacja stanowi wartościowe narzędzie dydaktyczne, demonstrując:

- Strukturę SVO typową dla języka angielskiego
- Zasady odmiany czasowników
- Tworzenie form pytających i przeczących
- Zgodność osoby i liczby między podmiotem a orzeczeniem

6.3 Projekt 3 - Analiza Semantyczna

6.3.1 Metodologia

Projekt wykorzystuje zaawansowane techniki NLP do analizy relacji semantycznych:

1. **Preprocessing i POS tagging:**
    - Tokenizacja korpusu 100,000 słów
    - Tagowanie części mowy przy użyciu modelu spaCy `en_core_web_sm`
    - Ekstrakcja rzeczowników (NOUN), przymiotników (ADJ) i czasowników (VERB)
2. **Analiza współwystępowania:**
    - Identyfikacja bigramów przymiotnik-rzeczownik
    - Identyfikacja bigramów czasownik-rzeczownik
    - Zliczanie częstotliwości każdej pary
3. **Budowa grafów dwudzielnych:**
    - Selekcja top 100 przymiotników i top 100 czasowników
    - Selekcja top 100 rzeczowników
    - Utworzenie grafów bipartite pokazujących wszystkie możliwe połączenia
4. **System kolorowania:**
    - **Czerwony:** 0 wystąpień (brak kolokacji)
    - **Żółty:** 1 wystąpienie (rzadka kolokacja)
    - **Zielony:** 2-10 wystąpień (częsta kolokacja)
    - **Niebieski:** >10 wystąpień (bardzo częsta kolokacja)

6.3.2 Wyniki i interpretacja

Grafy dwudzielne:

Wizualizacje grafowe przedstawiają sieć połączeń między częściami mowy. Kolory krawędzi pozwalają szybko zidentyfikować:

- Typowe kolokacje (np. "beautiful house", "read book")
- Rzadkie, ale możliwe kombinacje
- Brak naturalnego współwystępowania niektórych par

Macierze heatmap:

Wizualizacje w formie macierzy cieplnych (heatmap) przedstawiają wszystkie 10,000 możliwych kombinacji (100×100) dla każdej pary części mowy. Intensywność koloru odpowiada częstotliwości występowania danej kolokacji w korpusie.

Listy połączeń:

Programy generują szczegółowe pliki tekstowe (`lista_przymiotnik_rzeczownik.txt` i `lista_czasownik_rzeczownik.txt`), które dla każdego przymiotnika/czasownika wymieniają wszystkie rzeczowniki, z którymi występuje w korpusie, wraz z liczbą wystąpień.

6.3.3 Zastosowania praktyczne

Wyniki analizy mogą być wykorzystane do:

- **Walidacji poprawności tekstów** - sprawdzanie, czy dana kolokacja jest naturalna
- **Wspomagania tłumaczeń** - identyfikacja typowych związków frazeologicznych
- **Generowania języka naturalnego** - wybór najbardziej prawdopodobnych kombinacji słów
- **Analizy stylu** - porównywanie częstotliwości kolokacji między różnymi korpusami

6.3.4 Wygenerowane pliki

- `graf_przymiotnik_rzeczownik.png` - graf dwudzielny ADJ-NOUN
- `graf_czasownik_rzeczownik.png` - graf dwudzielny VERB-NOUN
- `macierz_przymiotnik_rzeczownik.png` - heatmap ADJ-NOUN (100×100)
- `macierz_czasownik_rzeczownik.png` - heatmap VERB-NOUN (100×100)
- `lista_przymiotnik_rzeczownik.txt` - szczegółowa lista wszystkich połączeń
- `lista_czasownik_rzeczownik.txt` - szczegółowa lista wszystkich połączeń

---

7. Zastosowane technologie i narzędzia

7.1 Języki programowania

- **Python 3.8+** - główny język implementacji wszystkich projektów

7.2 Biblioteki NLP

NLTK (Natural Language Toolkit):

- Tokenizacja tekstu
- POS tagging (Projekt 1)
- Przetwarzanie korpusów

spaCy:

- Zaawansowane POS tagging (Projekt 3)
- Model `en_core_web_sm` dla języka angielskiego
- Szybka i wydajna analiza dużych korpusów

inflect i lemminflect:

- Morfologia języka angielskiego (Projekt 2)
- Odmiana przez osoby i liczby
- Dobór przedimków a/an

7.3 Wizualizacja danych

matplotlib:

- Generowanie wykresów (prawo Zipfa)
- Tworzenie heatmap (macierze współwystępowania)

- Eksport do formatów graficznych (PNG)

NetworkX:

- Budowa i analiza grafów
- Wizualizacja grafów sąsiedztwa i grafów dwudzielnych
- Algorytmy grafowe

7.4 Framework webowy

Streamlit:

- Szybkie tworzenie interaktywnych aplikacji webowych
- Automatyczne odświeżanie interfejsu
- Proste komponenty UI (selectbox, radio, text)

7.5 Analiza danych

pandas:

- Manipulacja danymi tabelarycznymi
- Eksport do CSV
- Agregacja i grupowanie danych

NumPy:

- Operacje na macierzach
- Obliczenia numeryczne
- Wydajne przetwarzanie dużych zbiorów danych

---

8. Wnioski i podsumowanie

8.1 Osiągnięte cele

Projekt z powodzeniem zrealizował wszystkie założone cele:

1. **Weryfikacja teoretyczna:** Potwierdzono empirycznie prawo Zipfa na rzeczywistym korpusie językowym, co pokazuje uniwersalność praw statystycznych w języku naturalnym.
2. **Praktyczna implementacja:** Stworzono działające narzędzia NLP, które mogą mieć zastosowanie edukacyjne i praktyczne.
3. **Wizualizacja złożonych danych:** Skutecznie wizualizowano relacje semantyczne między tysiącami par słów, co umożliwia intuicyjne zrozumienie struktury języka.
4. **Interaktywność:** Generator zdań SVO dostarcza użytkownikowi natychmiastowego feedback'u, co znacząco podnosi wartość edukacyjną narzędzia.

8.2 Kluczowe obserwacje

Prawo Zipfa i ekonomia języka: Analiza korpusu wyraźnie pokazała, że język naturalny charakteryzuje się wysoką redundancją - niewielka liczba słów (około 20-30% unikalnych słów) pokrywa ogromną większość rzeczywistego użycia. Ma to głębokie implikacje dla kompresji tekstu, modelowania języka i budowy słowników.

Złożoność gramatyki: Implementacja generatora zdań SVO unaoczniła, jak wiele reguł i wyjątków istnieje nawet w pozornie prostej strukturze zdaniowej. Obsługa czasowników nieregularnych, zgodności osoby i liczby oraz tworzenie pytań i przeczeń wymaga rozbudowanej logiki.

Bogactwo kolokacji: Analiza semantyczna ujawniła, że choć teoretycznie możliwych jest 10,000 kombinacji (100×100), w praktyce tylko niewielka część z nich występuje w naturalnym języku. To pokazuje, że język nie jest przypadkowym zestawieniem słów, ale rządzi się silnymi preferencjami kolokacyjnymi.

8.3 Trudności i wyzwania

1. **Wydajność obliczeniowa:** Przetwarzanie korpusu 100,000 słów i generowanie grafów z tysięcami węzłów wymagało optymalizacji kodu i odpowiednich struktur danych.
2. **Jakość tagowania POS:** Różne taggery (NLTK vs. spaCy) dają nieco różne wyniki, co pokazuje, że automatyczne tagowanie wciąż nie jest zadaniem w pełni rozwiązanym.
3. **Obsługa wyjątków gramatycznych:** Implementacja wszystkich form nieregularnych czasowników w języku angielskim wymaga rozbudowanych słowników i specjalnej logiki.

8.4 Możliwości rozwoju

Projekt 1:

- Rozszerzenie analizy o n-gramy (trigramy, 4-gramy)
- Implementacja algorytmów wykrywania kolokacji (PMI, t-score)
- Porównanie wielu korpusów różnych gatunków

Projekt 2:

- Dodanie większej liczby struktur zdaniowych (np. zdania złożone)
- Obsługa innych czasów (Present Perfect, Past Continuous, etc.)
- Rozszerzenie o język polski
- Integracja z API do sprawdzania poprawności gramatycznej

Projekt 3:

- Analiza trigramów (przymiotnik-rzeczownik-czasownik)
- Identyfikacja idiomów i frazeologizmów
- Budowa modeli probabilistycznych do przewidywania kolokacji
- Analiza kontrastywna (porównanie różnych korpusów)

8.5 Wartość dydaktyczna

Realizacja projektu pozwoliła na praktyczne opanowanie:

- Narzędzi NLP (NLTK, spaCy)

- Technik wizualizacji danych językowych
- Metodologii przetwarzania i analizy korpusów
- Implementacji reguł gramatycznych w kodzie
- Budowy interaktywnych aplikacji webowych

Połączenie teorii z praktyką w trzech różnych obszarach NLP (statystyka, generowanie, semantyka) dało kompleksowy obraz możliwości i wyzwań przetwarzania języka naturalnego.

---

9. Bibliografia i źródła

Biblioteki i frameworki:

- Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media. (NLTK)
- Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*.
- Streamlit Inc. (2019-2026). *Streamlit Documentation*. https://docs.streamlit.io/

Teoria i algorytmy:

- Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley.
- Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.
- Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed. draft). https://web.stanford.edu/~jurafsky/slp3/

Dokumentacja techniczna:

- NLTK Project: https://www.nltk.org/
- spaCy Documentation: https://spacy.io/
- NetworkX Documentation: https://networkx.org/
- Matplotlib Documentation: https://matplotlib.org/

Korpusy i zasoby:

- Korpus testowy (100,000 słów) - dane własne projektu

Struktura projektu

```
Przetwarzanie_Jezyka_Naturalnego/
│
├── README.md                          # Niniejsze sprawozdanie
│
├── projekt_1/                         # Analiza Korpusu
│   ├── corpus/
│   │   └── corpus.txt                # Korpus 100k słów
│   ├── analiza_korpusu.py            # Program główny
│   ├── requirements.txt              # Zależności
│   ├── README.md                     # Dokumentacja projektu
│   ├── analiza_czestotliwosci.csv    # Wynik (generowany)
```

```
│   ├── wykres_zipfa.png            # Wynik (generowany)
│   └── graf_sasiedztwa.png         # Wynik (generowany)
│
├── projekt_2/                       # Generator SVO
│   ├── app.py                      # Aplikacja Streamlit
│   ├── data.json                   # Baza słów
│   ├── requirements.txt            # Zależności
│   └── README.md                   # Dokumentacja projektu
│
└── projekt_3/                       # Analiza Semantyczna
    ├── corpus/
    │   └── corpus.txt              # Korpus 100k słów
    ├── analiza_semantyczna.py      # Program główny
    ├── requirements.txt            # Zależności
    ├── README.md                   # Dokumentacja projektu
    ├── graf_przymiotnik_rzeczownik.png       # Wynik (generowany)
    ├── graf_czasownik_rzeczownik.png         # Wynik (generowany)
    ├── macierz_przymiotnik_rzeczownik.png    # Wynik (generowany)
    ├── macierz_czasownik_rzeczownik.png      # Wynik (generowany)
    ├── lista_przymiotnik_rzeczownik.txt      # Wynik (generowany)
    └── lista_czasownik_rzeczownik.txt        # Wynik (generowany)
```
