Sprawozdanie Lab 2: Skuteczne promptowanie modeli LLM w Ollama

Streszczenie

W laboratorium przetestowałem 8 różnych zastosowań promptowania LLM:

- NER, analiza sentymentu, ekstrakcja relacji
- Klasyfikacja tematyczna z porównaniem strategii
- Streszczenie i QA z kontekstem
- Praktyczny projekt asystenta turystycznego

Najważniejsze wnioski: few-shot poprawia wyniki (chociaż w poniższych zadaniach nieznacznie, bo o pare procent zawzyczaj, natomiast warto by było przetestować znacznie trudniejsze przypadki z trudniejszym kontekstem), modele mają trudności z liczeniem i precyzyjnym cytowaniem, polski model Bielik częściej halucynuje.

In [1]:
```
import json
import ollama
from pydantic import BaseModel
```

Zadanie 7.1: NER (Named Entity Recognition)

In [2]:
```
SYSTEM = "Jesteś precyzyjnym ekstraktorem nazw własnych. Analizuj tekst i klasyfikuj jednostki według podanych kategorii NER."

prompt = """Znajdź i sklasyfikuj wszystkie jednostki NER w tekście według poniższych kategorii:
- PERSON: nazwani bohaterowie
- PRODUCT: produkty, towary wytworzone przez człowieka
- ART: tytuły gier, filmów, książek, dzieł
- EVENT: wydarzenia, bitwy, wojny
- OBJECT: zamknięte przestrzenie, przedmioty
- BRAND: marki, firmy, producenci
- NORP: narodowości, grupy kulturowe
- DATE-PERIOD: przedziały czasu, pory

Zwróć wynik w formacie JSON."""

text = """
Cyberpunk 2077 jest fabularną grą akcji, zawierającą elementy strzelanek pierwszoosobowych. Gracz wciela się w V, którego płeć, głos, wygląd (twarz, fryzurę, sylwetkę, modyfikacje
Night City jest amerykańskim megamiastem w Wolnym Stanie Kalifornia Północna, kontrolowanym przez korporacje, w którym nie obowiązują prawa krajowe i stanowe. Ogarnięte jest wojną
"""

output_format = """
{
  "PERSON": ["imiona bohaterów"],
  "PRODUCT": ["produkty, towary"],
  "ART": ["tytuły gier, filmów"],
  "EVENT": ["wydarzenia, bitwy"],
  "OBJECT": ["przedmioty, bronie"],
  "BRAND": ["marki, firmy"],
  "NORP": ["narodowości, grupy"],
  "DATE-PERIOD": ["okresy czasu"]
}
"""

response = ollama.chat(
    model='gemma2:2b',
    messages=[
        {"role": "system", "content": SYSTEM},
        {"role": "user", "content": f"{prompt}\n\nTekst:\n{text}\n\nFormat:\n{output_format}"}
    ],
    format="json",
    options={"temperature": 0}
)

print("=== WYNIK NER - CYBERPUNK 2077 ===")
print(response['message']['content'])

response_pl = ollama.chat(
    model='SpeakLeash/bielik-1.5b-v3.0-instruct:Q8_0',
    messages=[
        {"role": "system", "content": SYSTEM},
        {"role": "user", "content": f"{prompt}\n\nTekst:\n{text}\n\nFormat:\n{output_format}"}
    ],
    format="json",
    options={"temperature": 0}
)

print("=== WYNIK NER - CYBERPUNK 2077 (Bielik 1.5B) ===")
print(response_pl['message']['content'])
```

```
=== WYNIK NER - CYBERPUNK 2077 ===
{
  "PERSON": [
    "V",
    "Psycho Squad",
    "Trauma Team"
  ],
  "PRODUCT": [
    "bronie",
    "cyberwszczepy",
    "robotyka",
    "kosmetyczne poprawek",
    "bronie palne",
    "pociski",
    "broń krótką",
    "strzelbami",
    "karabinami snajperskimi",
    "bronie do walki w zwarciu",
    "power (rykoszetująca)",
    "tech (przebijająca ściany i przeciwników)",
    "smart (z naprowadzanymi pociskami)"
  ],
  "ART": [
    "Cyberpunk 2077",
    "Night City"
  ],
  "EVENT": [
    "wojna gangów",
    "walka o dominację",
    "napaść"
  ],
  "OBJECT": [
    "osłony",
    "robotyka",
    "bronie palne",
    "cyberwszczepy",
    "pociski",
    "broń krótką",
    "strzelbami",
    "karabinami snajperskimi",
    "bronie do walki w zwarciu",
    "power (rykoszetująca)",
    "tech (przebijająca ściany i przeciwników)",
    "smart (z naprowadzanymi pociskami)"
  ],
  "BRAND": [
    "Night City",
    "korporacje",
    "wojsko",
    "Psycho Squad",
    "Trauma Team"
  ],
  "NORP": [
    "Ameryka",
    "Wolny Stany Kalifornia Północna"
  ],
  "DATE-PERIOD": [
    "okresy czasu"
  ]
}
=== WYNIK NER - CYBERPUNK 2077 (Bielik 1.5B) ===
{
  "PERSON": ["V", "Ripperdoc", "Trauma Team", "Psycho Squad"],
  "PRODUCT": ["Cyberpunk 2077", "cyberwszczepy", "bronie do walki w zwarciu", "bronie palna", "bronie dystansowe", "bronie ogłuszające", "bronie ulepszające skradanie się", "bronie
elektryczne", "bronie chemiczne", "bronie termiczne"],
  "ART": ["Cyberpunk 2077: Night City", "Cyberpunk 2077: Night City: Entropic Era", "Cyberpunk 2077: Night City: Neomilitary Era", "Cyberpunk 2077: Night City: Neokitchen Era"],
  "EVENT": ["wojna gangów", "konkurencja o dominację", "bezdomność", "cybermodyfikacje", "przemoc"],
  "OBJECT": ["osłony", "celowanie", "bieganie", "ślizgi", "skoki", "podwójne skoki", "broń do walki w zwarciu", "broń palna", "broń dystansowa", "broń ogłuszająca", "broń ulepszają
ca skradanie się"],
  "BRAND": ["Cyberpunk 2077", "Ripperdoc", "Trauma Team", "Psycho Squad"],
  "NORP": ["Kalifornia Północna", "Wolny Stan Kalifornia Północna"],
  "DATE-PERIOD": ["XXI wiek", "XXII wiek"]
}
```

Wnioski:

Model gemma2:2b wykazał dość trafnie przyporządkował i wykestrahował poszczególne nazwy własne, chociaż z niektórymi rzeczami miał trudności. "Psycho Squad", "Trauma Team" to są raczej frakcje lub grupy, a nie pojedyncze osoby czy organizacje, model podał je jako osoby. Date-period podaje jako "okres czasu", co jest poprawne, ale nie jest to standardowa kategoria NER, w tej kateogrii brakuje daty 2077. W Brand podaje "wojsko" czy też "korporacje" co jest zbyt ogólne. Źle zrozumiał nazwe własną "Object" jako rzecz/obiekt a nie przestrzeń w której odbywa się akcja gry/przebywają bohaterowie, co pokazuje, że model ma trudności z kontekstem. Ogólnie jednak model radzi sobie całkiem dobrze z rozpoznawaniem nazw własnych w tekście.

Bielik znacznie gorzej, zaczynał halucynować i podawać różne nieistniejące w tekście nazwy takie jak Cyberpunk 2077: Night City: Entropic Era itd. Podał w "PERSON" nazwe "Ripperdoc" co jest technicznie poprawne, chociaż nie jest to imię bohatera, a raczej profesji. W date-period halucynował. Polski model poradził sobie gorzej, mniej poprawnych przynależności do poszczeólnych kategorii jak i więcej halucynacji.

Źródło: https://literacka.com.pl/2021/08/analiza-ner-czyli-o-czytaniu-tekstu-za-pomoca-sztucznej-inteligencji/

Zadanie 7.2: Analiza sentymentu

In [3]:
```python
SYSTEM = "Jesteś precyzyjnym analizatorem wydźwięku tekstu. Zwracaj wyłącznie poprawny JSON według podanej specyfikacji."

def analyze_sentiment(text, model_name='gemma2:2b'):
    prompt = f"""Określ wydźwięk (pozytywny/negatywny/neutralny) poniższego tekstu.
    Zwróć JSON z kluczami: "sentiment", "confidence" (1-10), "evidence" (najważniejsze słowa/frazy).

    Tekst: {text}
    """

    response = ollama.chat(
        model=model_name,
        messages=[
            {"role": "system", "content": SYSTEM},
            {"role": "user", "content": prompt}
        ],
        format="json",
        options={"temperature": 0}
    )
    return response['message']['content']
```

```python
# Przykład 1: Pozytywna recenzja restauracji
positive_review = """
Restauracja "Smaki Polski" to prawdziwa perełka! Obsługa była niezwykle uprzejma i profesjonalna.
Pierogi robione na miejscu smakowały jak u babci, a kotlet schabowy był idealnie chrupiący.
Atmosfera ciepła i przytulna, ceny rozsądne. Z całego serca polecam!
"""

# Przykład 2: Sarkastyczna recenzja (negatywna ukryta w ironii)
sarcastic_review = """
Wow, ta restauracja to prawdziwe "arcydzieło"! Czekanie 45 minut na zimne jedzenie to była
"fantastyczna" przygoda. Obsługa tak "profesjonalna", że kelner zapomniał o naszym stoliku.
A te ceny! Naprawdę "świetny" stosunek jakości do ceny - płacisz za restaurację,
a dostajesz jakość fast-foodu. Zdecydowanie "polecam" wszystkim, którzy lubią
marnować czas i pieniądze.
"""

# Przykład 3: Neutralny opis faktyczny
neutral_text = """
Restauracja mieści się przy ulicy Głównej 15. Oferuje dania kuchni polskiej i międzynarodowej.
Godziny otwarcia: poniedziałek-piątek 12:00-22:00, sobota-niedziela 14:00-23:00.
Menu dostępne online i w lokalu. Możliwość rezerwacji stolików telefonicznych.
"""

print("=== ANALIZA SENTYMENTU - PRZYKŁADY WŁASNE ===\n")

# Test na różnych modelach
models = ['gemma2:2b', 'llama3.1:8b']

for model in models:
    print(f"--- MODEL: {model} ---")

    try:
        # Pozytywny
        result1 = analyze_sentiment(positive_review, model)
        print("POZYTYWNY:")
        print(json.dumps(json.loads(result1), indent=2, ensure_ascii=False))

        # Sarkastyczny
        result2 = analyze_sentiment(sarcastic_review, model)
        print("\nSARKASTYCZNY:")
        print(json.dumps(json.loads(result2), indent=2, ensure_ascii=False))

        # Neutralny
        result3 = analyze_sentiment(neutral_text, model)
        print("\nNEUTRALNY:")
        print(json.dumps(json.loads(result3), indent=2, ensure_ascii=False))

    except Exception as e:
        print(f"Błąd dla modelu {model}: {e}")

    print("\n" + "="*50 + "\n")

# Dodatkowy test - czy model rozpoznaje sarkasm
print("=== TEST ROZPOZNAWANIA SARKAZMU ===")

sarcasm_prompt = """
Przeanalizuj poniższy tekst pod kątem:
1. Dosłownego znaczenia słów
2. Prawdopodobnego sarkazmu/ironii
3. Rzeczywistego wydźwięku

Zwróć JSON: {
  "literal_sentiment": "pozytywny/negatywny/neutralny",
  "detected_sarcasm": true/false,
  "actual_sentiment": "pozytywny/negatywny/neutralny",
  "sarcasm_indicators": ["lista wskaźników sarkazmu"]
}

Tekst: """ + sarcastic_review

response = ollama.chat(
    model='gemma2:2b',
    messages=[
        {"role": "system", "content": "Jesteś ekspertem od analizy stylistycznej tekstu, specjalizującym się w wykrywaniu sarkazmu i ironii."},
        {"role": "user", "content": sarcasm_prompt}
    ],
    format="json",
    options={"temperature": 0}
)

print("ANALIZA SARKAZMU:")
print(json.dumps(json.loads(response['message']['content']), indent=2, ensure_ascii=False))
```

```
=== ANALIZA SENTYMENTU - PRZYKŁADY WŁASNE ===

--- MODEL: gemma2:2b ---
POZYTYWNY:
{
  "sentiment": "positive",
  "confidence": 9,
  "evidence": [
    "perełka",
    "uprzejma i profesjonalna",
    "smakły jak u babci",
    "idealnie chrupiący",
    "ciepła i przytulna",
    "z całego serca polecam!"
  ]
}

SARKASTYCZNY:
{
  "sentiment": "positive",
  "confidence": 9,
  "evidence": [
    "arcydzieło",
    "fantastyczna przygoda",
    "profesjonalna",
    "świetny",
    "polecam",
    "marnować czas i pieniądze"
  ]
}

NEUTRALNY:
{
  "sentiment": "neutral",
  "confidence": 7,
  "evidence": [
    "Restauracja mieści się przy ulicy Głównej 15.",
    "Oferuje dania kuchni polskiej i międzynarodowej.",
    "Godziny otwarcia: poniedziałek-piątek 12:00-22:00, sobota-niedziela 14:00-23:00.",
    "Menu dostępne online i w lokalu.",
    "Możliwość rezerwacji stolików telefonicznych."
  ]
}

==================================================

--- MODEL: llama3.1:8b ---
POZYTYWNY:
{
  "sentiment": "pozytywny",
  "confidence": 10,
  "evidence": [
    "prawdziwa perełka",
    "niezwykle uprzejma",
    "profesjonalna",
    "smakowały jak u babci",
    "chrupiący",
    "ciepła i przytulna",
    "ceny rozsądne"
  ]
}

SARKASTYCZNY:
{
  "sentiment": "negatywny",
  "confidence": 8,
  "evidence": [
    "kelner zapomniał o naszym stoliku",
    "ceny",
    "marnować czas i pieniądze"
  ]
}

NEUTRALNY:
{
  "sentiment": "neutralny",
  "confidence": 8,
  "evidence": [
    "oferuje dania kuchni polskiej i międzynarodowej",
    "menu dostępne online i w lokalu"
  ]
}

==================================================

=== TEST ROZPOZNAWANIA SARKAZMU ===
ANALIZA SARKAZMU:
{
  "literal_sentiment": "negatywny",
  "detected_sarcasm": true,
  "actual_sentiment": "negatywny",
  "sarcasm_indicators": [
    "45 minut na zimne jedzenie to była 'fantastyczna' przygoda.",
    "profesjonalna",
    "zapomniał o naszym stoliku.",
    "'świetny' stosunek jakości do ceny - płacisz za restaurację, a dostajesz jakość fast-foodu.",
    "polecam wszystkim, którzy lubią marnować czas i pieniądze."
  ]
}
```

Wnioski:

Model nie radzi sobie dobrze z analizą sentymentu w języku polskim, co może wynikać z mniejszej ilości danych treningowych w tym języku. Pomyłił sarkastyczny komentarzy jako bardzo pozytywny, co pokazuje, że ma trudności z rozpoznawaniem niuansów językowych. W pozostałych 2 przypadkach poradził sobie dobrze, poprawnie klasyfikując sentyment jako pozytywny i neutralny. Model llama3.1 już we wszystkich 3 przypadkach poradził sobie dobrze, poprawnie klasyfikując sentyment jako pozytywny, neutralny i negatywny. Natomiast udało się dodać odpowiedni prompt do modelu gemma2:2b, twierdząc, że jest to model który ma sobie dobrze radzić w przypadku sarkastycznych wypowiedzi i w ten sposób poprawić jego skuteczność w tym zakresie. W ten sposób model gemma2:2b poradził sobie dobrze we wszystkich 3 przypadkach poprawnie je klasyfikując.

Zadanie 7.3: Ekstrakcja relacji

In [35]:
```
SYSTEM = "Jesteś precyzyjnym analizatorem tekstu. Znajdź TYLKO relacje PRACUJE_W i MIESZKA_W. Zwróć sformatowaną tablicę JSON."

# Few-shot przykłady do trenowania modelu
shots = [
    {
        "role": "user",
        "content": "Tekst: \"Maria Nowak pracuje w banku PKO BP jako analityk finansowy.\""
    },
    {
        "role": "assistant",
        "content": '[{"person": "Maria Nowak", "relation": "PRACUJE_W", "target": "PKO BP"}]'
    },
    {
        "role": "user",
        "content": "Tekst: \"Tomasz Kowalski mieszka w Krakowie i pracuje w firmie Google.\""
    },
    {
        "role": "assistant",
        "content": '[{"person": "Tomasz Kowalski", "relation": "MIESZKA_W", "target": "Kraków"}, {"person": "Tomasz Kowalski", "relation": "PRACUJE_W", "target": "Google"}]'
    },
    {
        "role": "user",
        "content": "Tekst: \"Jan Nowak mieszka w Warszawie. Obecnie pracuje w Microsoft, ale wcześniej był zatrudniony w IBM.\""
    },
    {
        "role": "assistant",
        "content": '[{"person": "Jan Nowak", "relation": "MIESZKA_W", "target": "Warszawa"}, {"person": "Jan Nowak", "relation": "PRACUJE_W", "target": "Microsoft"}]'
    }
]

# Przykładowe teksty do analizy
test_texts = [
    """
    Dr Aleksandra Wiśniewska jest profesorem na Uniwersytecie Jagiellońskim w Krakowie.
    Mieszka w Warszawie, ale dojeżdża do pracy. Wcześniej pracowała w Institute of Technology w Bostonie,
    gdzie mieszkała przez 5 lat.
    """,

    """
    Michał Zieliński to programista w firmie CD Projekt. Mieszka w Gdańsku, ale pracuje
    zdalnie. Jego żona Anna Zielińska pracuje w szpitalu miejskim w Gdańsku jako lekarka.
    """,

    """
    Startup TechnoVision został założony przez Pawła Nowaka, który mieszka w Poznaniu.
    Firma ma biura w Warszawie i Wrocławiu. Paweł wcześniej pracował w Microsoft Poland.
    """
]

print("=== EKSTRAKCJA RELACJI - PRZYKŁADY WŁASNE ===\n")

# Test few-shot z przykładami
for text in test_texts:
    print(f"--- TEKST DO ANALIZY ---\n{text.strip()}\n")

    shots_with_test = shots + [
        {"role": "user", "content": f"Tekst: \"{text}\""}
    ]

    response = ollama.chat(
        model='gemma2:2b',
        messages=[
            {"role": "system", "content": SYSTEM},
            *shots_with_test
        ]
    )
    print(response['message']['content'])
```

```
=== EKSTRAKCJA RELACJI - PRZYKŁADY WŁASNE ===

--- TEKST DO ANALIZY ---
Dr Aleksandra Wiśniewska jest profesorem na Uniwersytecie Jagiellońskim w Krakowie.
    Mieszka w Warszawie, ale dojeżdża do pracy. Wcześniej pracowała w Institute of Technology w Bostonie,
    gdzie mieszkała przez 5 lat.

```json
[{"person": "Dr Aleksandra Wiśniewska", "relation": "MIESZKA_W", "target": "Kraków"}, {"person": "Dr Aleksandra Wiśniewska", "relation": "MIESZKA_W", "target": "Warszawa"}]
```


--- TEKST DO ANALIZY ---
Michał Zieliński to programista w firmie CD Projekt. Mieszka w Gdańsku, ale pracuje
    zdalnie. Jego żona Anna Zielińska pracuje w szpitalu miejskim w Gdańsku jako lekarka.

[{"person": "Michał Zieliński", "relation": "PRACUJE_W", "target": "CD Projekt"}, {"person": "Michał Zieliński", "relation": "MIESZKA_W", "target": "Gdańsk"}, {"person": "Anna Ziel
ińska", "relation": "PRACUJE_W", "target": "szpital miejski w Gdańsku"}, {"person": "Anna Zielińska", "relation": "MIESZKA_W", "target": "Gdańsk"}]

--- TEKST DO ANALIZY ---
Startup TechnoVision został założony przez Pawła Nowaka, który mieszka w Poznaniu.
    Firma ma biura w Warszawie i Wrocławiu. Paweł wcześniej pracował w Microsoft Poland.

```json
[{"person": "Pawel Nowak", "relation": "MIESZKA_W", "target": "Poznań"}, {"person": "Pawel Nowak", "relation": "PRACUJE_W", "target": "Microsoft Poland"}, {"person": "Pawel Nowak",
"relation": "PRACUJE_W", "target": "Startup TechnoVision"}]
```
```

In [36]:
```
print ("=== EKTRAKCJA RELACJI - TEST NA MODELU BIELIK 1.5B ===\n")

for text in test_texts:
    print(f"--- TEKST DO ANALIZY ---\n{text.strip()}\n")

    shots_with_test = shots + [
        {"role": "user", "content": f"Tekst: \"{text}\""}
    ]
```

```
    response = ollama.chat(
        model='SpeakLeash/bielik-1.5b-v3.0-instruct:Q8_0',
        messages=[
            {"role": "system", "content": SYSTEM},
            *shots_with_test
        ],
        options={"temperature": 0}
    )
    print(response['message']['content'])
```

```
=== EKTRAKCJA RELACJI - TEST NA MODELU BIELIK 1.5B ===

--- TEKST DO ANALIZY ---
Dr Aleksandra Wiśniewska jest profesorem na Uniwersytecie Jagiellońskim w Krakowie.
    Mieszka w Warszawie, ale dojeżdża do pracy. Wcześniej pracowała w Institute of Technology w Bostonie,
    gdzie mieszkała przez 5 lat.

[{"person": "Dr Aleksanara Wiśniewska", "related_persons": ["Institute of Technology in Boston"], "related_place": "Kraków, Warszawa"}]
--- TEKST DO ANALIZY ---
Michał Zieliński to programista w firmie CD Projekt. Mieszka w Gdańsku, ale pracuje
    zdalnie. Jego żona Anna Zielińska pracuje w szpitalu miejskim w Gdańsku jako lekarka.

[{"person": "Michał Zieliński", "relation": "MIESZKA_W", "target": "CD Projekt"}, {"personal": "Anna Zielińska", "related": "PROCEDENT", "target": "Gdańsk", "type": "lekarz"}]
--- TEKST DO ANALIZY ---
Startup TechnoVision został założony przez Pawła Nowaka, który mieszka w Poznaniu.
    Firma ma biura w Warszawie i Wrocławiu. Paweł wcześniej pracował w Microsoft Poland.

[{"person": "Paweł Nowak", "relation": "MIESZKA_W", "target": "Poznań"}, {"personal": "TechnoVision", "related": "Startup", "target": "Warszawa"}, {"personal": "TechnoVision", "rel
ated": "Startup", "target": "Wrocław"}]
```

Wnioski

Model początkowo miał trudności z poprawnym zidentyfikowaniem relacji, podawał i tworzył własne relacje, które istniały w tekście, nie zawsze poprawnie je klasyfikując. Musiałem wymusić w promptcie, aby model skupił się tylko na relacjach MIESZKA_W oraz PRACUJE_W. Po dodaniu tego do promptu model radził sobie już lepiej, identyfikując tylko interesujące nas relacje. Całkiem nieźle klasyfikował relacje, chociaż w 1 przypadku nie podał w ogóle relacji PRACUJE_W oraz podał 2 relacje MIESZKA_W zamiast jednej. Widocznie model ma trudności z zrozumieniem czasu przeszłego i teraźniejszego w kontekście relacji.

Bielik natomiast średnio sobie radził, niektóre relacje rzeczywiście były poprawne, ale wymyślał własne klucze w jsonie, własne nazwy i relacje, często mało zrozumiale.

Zadanie 7.4: Klasyfikacja tematyczna

In [9]:
```
# Zbiór testowych tekstów z różnych kategorii
test_texts = [
    "Nowy mikroskop pozwala obserwować pojedyncze atomy w czasie rzeczywistym.",
    "Robert Lewandowski strzelił hat-tricka w meczu przeciwko Bayernowi Monachium.",
    "Parlament przyjął ustawę o reformie systemu podatkowego większością głosów.",
    "Sztuczna inteligencja ChatGPT-4 osiągnęła przełomowe wyniki w testach językowych.",
    "Naukowcy z MIT odkryli nowy materiał przewodzący prąd w temperaturze pokojowej.",
    "Reprezentacja Polski awansowała do półfinału mistrzostw świata w siatkówce.",
    "Premier zapowiedział zwiększenie wydatków na ochronę zdrowia w przyszłym roku.",
    "Apple przedstawiło nowy procesor M3 z architekturą 3-nanometrową."
]

# Oczekiwane kategorie (dla weryfikacji)
expected_categories = [
    "nauka", "sport", "polityka", "technologia",
    "nauka", "sport", "polityka", "technologia"
]
```

In [10]:
```
# WARIANT 1: ZERO-SHOT
print("--- WARIANT 1: ZERO-SHOT ---")

SYSTEM_ZERO = (
    "Jesteś klasyfikatorem tematów. Przypisz tekst do JEDNEJ kategorii "
    "z zestawu: nauka, sport, polityka, technologia. "
    "Zwróć wyłącznie JSON: {\"kategoria\":\"...\"}."
)

zero_shot_results = []

for i, text in enumerate(test_texts):
    try:
        response = ollama.chat(
            model='gemma2:2b',
            messages=[
                {"role": "system", "content": SYSTEM_ZERO},
                {"role": "user", "content": f"Tekst: \"{text}\""}
            ],
            format="json",
            options={"temperature": 0}
        )

        result = json.loads(response['message']['content'])
        predicted = result.get('kategoria', 'BŁĄD')
        zero_shot_results.append(predicted)

        print(f"{i+1}. {text[:50]}...")
        print(f"   Przewidziano: {predicted} | Oczekiwano: {expected_categories[i]}")

    except Exception as e:
        print(f"Błąd: {e}")
        zero_shot_results.append('BŁĄD')
```

```
--- WARIANT 1: ZERO-SHOT ---
1. Nowy mikroskop pozwala obserwować pojedyncze atomy...
   Przewidziano: technologia | Oczekiwano: nauka
2. Robert Lewandowski strzelił hat-tricka w meczu prz...
   Przewidziano: sport | Oczekiwano: sport
3. Parlament przyjął ustawę o reformie systemu podatk...
   Przewidziano: polityka | Oczekiwano: polityka
4. Sztuczna inteligencja ChatGPT-4 osiągnęła przełomo...
   Przewidziano: technologia | Oczekiwano: technologia
5. Naukowcy z MIT odkryli nowy materiał przewodzący p...
   Przewidziano: technologia | Oczekiwano: nauka
6. Reprezentacja Polski awansowała do półfinału mistr...
   Przewidziano: sport | Oczekiwano: sport
7. Premier zapowiedział zwiększenie wydatków na ochro...
   Przewidziano: polityka | Oczekiwano: polityka
8. Apple przedstawiło nowy procesor M3 z architekturą...
   Przewidziano: technologia | Oczekiwano: technologia
```

In [11]:

```python
# WARIANT 2: FEW-SHOT
print("\n--- WARIANT 2: FEW-SHOT ---")

SYSTEM_FEW = "Klasyfikuj teksty do jednej z kategorii: nauka, sport, polityka, technologia."

few_shot_examples = [
    {
        "role": "user",
        "content": "Tekst: \"Badacze z Harvardu opracowali nową szczepionkę przeciwko grypie.\""
    },
    {
        "role": "assistant",
        "content": "{\"kategoria\":\"nauka\"}"
    },
    {
        "role": "user",
        "content": "Tekst: \"Cristiano Ronaldo zdobył bramkę w 90. minucie meczu.\""
    },
    {
        "role": "assistant",
        "content": "{\"kategoria\":\"sport\"}"
    },
    {
        "role": "user",
        "content": "Tekst: \"Sejm uchwalił budżet państwa na następny rok.\""
    },
    {
        "role": "assistant",
        "content": "{\"kategoria\":\"polityka\"}"
    },
    {
        "role": "user",
        "content": "Tekst: \"Google zaprezentowało nowy algorytm uczenia maszynowego.\""
    },
    {
        "role": "assistant",
        "content": "{\"kategoria\":\"technologia\"}"
    }
]

few_shot_results = []

for i, text in enumerate(test_texts):
    try:
        messages = [{"role": "system", "content": SYSTEM_FEW}] + few_shot_examples + [
            {"role": "user", "content": f"Tekst: \"{text}\""}
        ]

        response = ollama.chat(
            model='gemma2:2b',
            messages=messages,
            format="json",
            options={"temperature": 0}
        )

        result = json.loads(response['message']['content'])
        predicted = result.get('kategoria', 'BŁĄD')
        few_shot_results.append(predicted)

        print(f"{i+1}. {text[:50]}...")
        print(f"   Przewidziano: {predicted} | Oczekiwano: {expected_categories[i]}")

    except Exception as e:
        print(f"Błąd: {e}")
        few_shot_results.append('BŁĄD')
```

```
--- WARIANT 2: FEW-SHOT ---
1. Nowy mikroskop pozwala obserwować pojedyncze atomy...
   Przewidziano: nauka | Oczekiwano: nauka
2. Robert Lewandowski strzelił hat-tricka w meczu prz...
   Przewidziano: sport | Oczekiwano: sport
3. Parlament przyjął ustawę o reformie systemu podatk...
   Przewidziano: polityka | Oczekiwano: polityka
4. Sztuczna inteligencja ChatGPT-4 osiągnęła przełomo...
   Przewidziano: nauka, technologia | Oczekiwano: technologia
5. Naukowcy z MIT odkryli nowy materiał przewodzący p...
   Przewidziano: nauka | Oczekiwano: nauka
6. Reprezentacja Polski awansowała do półfinału mistr...
   Przewidziano: sport | Oczekiwano: sport
7. Premier zapowiedział zwiększenie wydatków na ochro...
   Przewidziano: polityka | Oczekiwano: polityka
8. Apple przedstawiło nowy procesor M3 z architekturą...
   Przewidziano: technologia | Oczekiwano: technologia
```

In [12]:

```python
# WARIANT 3: SEKCYJNY (RULES-INPUT-OUTPUT)
print("\n--- WARIANT 3: SEKCYJNY ---")

def classify_sectional(text):
    prompt = f"""
<RULES>
Przypisz tekst dokładnie do jednej z 4 kategorii:
- nauka: badania, odkrycia, eksperymenty, publikacje naukowe
- sport: mecze, zawody, wyniki sportowe, zawodnicy
- polityka: wybory, ustawy, rząd, partie, decyzje polityczne
- technologia: nowe urządzenia, oprogramowanie, IT, innowacje tech
</RULES>
```

```
<INPUT>
{text}
</INPUT>

<OUTPUT>
Zwróć JSON: {{"kategoria": "nauka/sport/polityka/technologia", "uzasadnienie": "krótkie uzasadnienie"}}
</OUTPUT>
"""

    response = ollama.chat(
        model='gemma2:2b',
        messages=[
            {"role": "system", "content": "Jesteś precyzyjnym klasyfikatorem. Postępuj zgodnie z instrukcjami w sekcjach."},
            {"role": "user", "content": prompt}
        ],
        format="json",
        options={"temperature": 0}
    )

    return response['message']['content']

sectional_results = []

for i, text in enumerate(test_texts):
    try:
        result_raw = classify_sectional(text)
        result = json.loads(result_raw)
        predicted = result.get('kategoria', 'BŁĄD')
        sectional_results.append(predicted)

        print(f"{i+1}. {text[:50]}...")
        print(f"   Przewidziano: {predicted} | Oczekiwano: {expected_categories[i]}")
        print(f"   Uzasadnienie: {result.get('uzasadnienie', 'Brak')}")

    except Exception as e:
        print(f"Błąd: {e}")
        sectional_results.append('BŁĄD')
```

```
--- WARIANT 3: SEKCYJNY ---
1. Nowy mikroskop pozwala obserwować pojedyncze atomy...
   Przewidziano: nauka | Oczekiwano: nauka
   Uzasadnienie: Tekst opisuje nowy mikroskop, który pozwala na obserwację pojedynczych atomy w czasie rzeczywistym. To dotyczy badań naukowych.
2. Robert Lewandowski strzelił hat-tricka w meczu prz...
   Przewidziano: sport | Oczekiwano: sport
   Uzasadnienie: Zawodnik strzelił hat-trick w meczu, co jest wydarzeniem sportowym.
3. Parlament przyjął ustawę o reformie systemu podatk...
   Przewidziano: polityka | Oczekiwano: polityka
   Uzasadnienie: W tekście opisano przyjęcie ustawy przez parlament, co jest elementem polityki.
4. Sztuczna inteligencja ChatGPT-4 osiągnęła przełomo...
   Przewidziano: nauka | Oczekiwano: technologia
   Uzasadnienie: Tekst opisuje osiągnięcia w dziedzinie sztucznej inteligencji, co jest tematem badań i odkryć w nauce.
5. Naukowcy z MIT odkryli nowy materiał przewodzący p...
   Przewidziano: nauka | Oczekiwano: nauka
   Uzasadnienie: Tekst opisuje odkrycie naukowe nowego materiału, co jest związane z badaniami i nauką.
6. Reprezentacja Polski awansowała do półfinału mistr...
   Przewidziano: sport | Oczekiwano: sport
   Uzasadnienie: Zawody w siatkówce, awans do półfinału mistrzostw świata.
7. Premier zapowiedział zwiększenie wydatków na ochro...
   Przewidziano: polityka | Oczekiwano: polityka
   Uzasadnienie: W tekście mowa jest o zwiększeniu wydatków na ochronę zdrowia, co jest tematem polityki
8. Apple przedstawiło nowy procesor M3 z architekturą...
   Przewidziano: nauka | Oczekiwano: technologia
   Uzasadnienie: Tekst opisuje nowy procesor M3 z architekturą 3-nanometrową, co jest tematem badań i innowacji w dziedzinie technologii.
```

In [13]:

```
# ANALIZA WYNIKÓW
print("\n=== ANALIZA PORÓWNAWCZA ===")

def calculate_accuracy(predicted, expected):
    correct = sum(1 for p, e in zip(predicted, expected) if p == e)
    total = len(expected)
    return correct / total if total > 0 else 0

accuracy_zero = calculate_accuracy(zero_shot_results, expected_categories)
accuracy_few = calculate_accuracy(few_shot_results, expected_categories)
accuracy_sectional = calculate_accuracy(sectional_results, expected_categories)

print(f"Dokładność Zero-shot: {accuracy_zero:.2%}")
print(f"Dokładność Few-shot: {accuracy_few:.2%}")
print(f"Dokładność Sekcyjna: {accuracy_sectional:.2%}")

# Tabela porównawcza
print("\n--- TABELA PORÓWNAWCZA ---")
print("Nr | Oczekiwane | Zero-shot | Few-shot | Sekcyjna")
print("-" * 50)

for i in range(len(test_texts)):
    expected = expected_categories[i]
    zero = zero_shot_results[i]
    few = few_shot_results[i]
    sect = sectional_results[i]

    print(f"{i+1:2d} | {expected:10s} | {zero:9s} | {few:8s} | {sect:8s}")

# Analiza błędów
print("\n--- ANALIZA BŁĘDÓW ---")
categories = ["nauka", "sport", "polityka", "technologia"]

for strategy_name, results in [("Zero-shot", zero_shot_results),
                               ("Few-shot", few_shot_results),
                               ("Sekcyjna", sectional_results)]:
    print(f"\n{strategy_name}:")
    errors = [(i, expected_categories[i], results[i])
              for i in range(len(results))
              if results[i] != expected_categories[i]]

    if errors:
        for idx, expected, predicted in errors:
            print(f"  Błąd {idx+1}: {expected} → {predicted}")
            print(f"    Tekst: {test_texts[idx][:60]}...")
    else:
        print("  Brak błędów!")

print(f"\n=== PODSUMOWANIE ===")
```

```
print(f"Najlepsza strategia: ", end="")
best_accuracy = max(accuracy_zero, accuracy_few, accuracy_sectional)
if best_accuracy == accuracy_sectional:
    print("Sekcyjna")
elif best_accuracy == accuracy_few:
    print("Few-shot")
else:
    print("Zero-shot")
```

```
=== ANALIZA PORÓWNAWCZA ===
Dokładność Zero-shot: 75.00%
Dokładność Few-shot: 87.50%
Dokładność Sekcyjna: 75.00%

--- TABELA PORÓWNAWCZA ---
Nr | Oczekiwane | Zero-shot | Few-shot | Sekcyjna
--------------------------------------------------
 1 | nauka      | technologia | nauka    | nauka
 2 | sport      | sport     | sport    | sport
 3 | polityka   | polityka  | polityka | polityka
 4 | technologia | technologia | nauka, technologia | nauka
 5 | nauka      | technologia | nauka    | nauka
 6 | sport      | sport     | sport    | sport
 7 | polityka   | polityka  | polityka | polityka
 8 | technologia | technologia | technologia | nauka

--- ANALIZA BŁĘDÓW ---

Zero-shot:
  Błąd 1: nauka → technologia
    Tekst: Nowy mikroskop pozwala obserwować pojedyncze atomy w czasie ...
  Błąd 5: nauka → technologia
    Tekst: Naukowcy z MIT odkryli nowy materiał przewodzący prąd w temp...

Few-shot:
  Błąd 4: technologia → nauka, technologia
    Tekst: Sztuczna inteligencja ChatGPT-4 osiągnęła przełomowe wyniki ...

Sekcyjna:
  Błąd 4: technologia → nauka
    Tekst: Sztuczna inteligencja ChatGPT-4 osiągnęła przełomowe wyniki ...
  Błąd 8: technologia → nauka
    Tekst: Apple przedstawiło nowy procesor M3 z architekturą 3-nanomet...

=== PODSUMOWANIE ===
Najlepsza strategia: Few-shot
```

Wnioski:

Jak się można było spodziewać najlepiej poradziła sobie strategia few-shot, gdzie model miał podane przykłady i mógł się na nich wzorować. Co ciekawe podał dwie kategorie w jednym przypadku (nauka, technologia), co pokazuje, że model potrafi rozumieć, że tekst może należeć do więcej niż jednej kategorii. W tym przypadku akurat poprawna była jedna kategoria (technologia), ale i tak jest to ciekawa obserwacja. Prawdopodobnie należało wymusić w promptcie, aby model podawał tylko jedną kategorię.

W pozostałych strategiach model radził sobie ciut gorzej, szczególnie w strategii zero-shot, gdzie nie miał żadnych przykładów do naśladowania. Widać, że dodanie przykładów (few-shot) znacząco poprawia skuteczność klasyfikacji tematycznej przez modele LLM. Natomiast ostatecznie pozostałe miały accuracy na poziomie 75%. W few-shot accuracy wyniosło 87.5%, a nawet 100% w najlepszym przypadku (gdybyśmy zaakceptowali dwie kategorie w jednym przypadku).

Ciekawą obserwacją jest to, że model często mylił kategorię "nauka" z "technologia", co może wynikać z bliskości tych tematów. Może to sugerować, że model ma trudności z rozróżnieniem tych dwóch kategorii na podstawie krótkich tekstów.

Ostatecznie przykłady były dość proste i oczywiste, brakowało trudniejszych przykładów.

Zadanie 7.5: Streszczenie i parafraza

In [15]:
```
SYSTEM = "Jesteś ekspertem od streszczania i parafrazowania tekstu. Zwracasz wyłącznie poprawny JSON."

text = """
Rewolucja w dziedzinie sztucznej inteligencji przyspiesza w zawrotnym tempie. Nowe modele językowe,
takie jak GPT-4 czy Claude, potrafią już nie tylko generować tekst, ale także rozumować, kodować
i rozwiązywać złożone problemy. Firmy technologiczne inwestują miliardy dolarów w rozwój AI,
przewidując, że będzie to kluczowa technologia następnej dekady.

Jednak rozwój sztucznej inteligencji niesie ze sobą również wyzwania. Eksperci ostrzegają przed
ryzykiem utraty miejsc pracy w wielu sektorach, problemami z dezinformacją oraz kwestiami
bezpieczeństwa cybernetycznego. Rządy na całym świecie pracują nad regulacjami, które miałyby
kontrolować rozwój AI, zachowując równowagę między innowacją a bezpieczeństwem publicznym.

W sektorze edukacji AI już teraz zmienia sposób nauczania i uczenia się. Spersonalizowane systemy
edukacyjne dostosowują się do indywidualnych potrzeb uczniów, podczas gdy nauczyciele korzystają
z narzędzi AI do tworzenia materiałów dydaktycznych i oceniania prac. Przyszłość edukacji będzie
prawdopodobnie opierać się na współpracy między człowiekiem a sztuczną inteligencją.
"""

print(f"ORYGINALNY TEKST ({len(text.split())} słów):")
print(text.strip())
print("\n" + "="*60 + "\n")

# PROMPT W SCHEMACIE SEKCYJNYM
prompt = f"""
<ROLE>
Jesteś specjalistą od przetwarzania tekstu. Tworzysz wysokiej jakości streszczenia i parafrazy.
</ROLE>

<RULES>
1. Krótkie streszczenie: maksymalnie 25 słów, zachowaj najważniejsze informacje
2. Długie streszczenie: maksymalnie 70 słów, uwzględnij kluczowe szczegóły
3. Parafraza: przepisz krótkie streszczenie innymi słowami, zachowaj sens
4. Zwróć wynik w formacie JSON
</RULES>

<INPUT>
{text}
</INPUT>

<OUTPUT>
{{
  "short_summary": "krótkie streszczenie (max 25 słów)",
  "long_summary": "długie streszczenie (max 70 słów)",
```

```
  "paraphrase": "parafraza krótkiego streszczenia"
}}
</OUTPUT>
"""

response = ollama.chat(
    model='gemma2:2b',
    messages=[
        {"role": "system", "content": SYSTEM},
        {"role": "user", "content": prompt}
    ],
    format="json",
    options={"temperature": 0}
)

result = json.loads(response['message']['content'])

print("=== WYNIKI STRESZCZANIA I PARAFRAZOWANIA ===\n")

print("KRÓTKIE STRESZCZENIE:")
print(f"→ {result['short_summary']}")
print(f"  Długość: {len(result['short_summary'].split())} słów\n")

print("DŁUGIE STRESZCZENIE:")
print(f"→ {result['long_summary']}")
print(f"  Długość: {len(result['long_summary'].split())} słów\n")

print("PARAFRAZA KRÓTKIEGO STRESZCZENIA:")
print(f"→ {result['paraphrase']}")
print(f"  Długość: {len(result['paraphrase'].split())} słów\n")
```

```
ORYGINALNY TEKST (140 słów):
Rewolucja w dziedzinie sztucznej inteligencji przyspiesza w zawrotnym tempie. Nowe modele językowe,
takie jak GPT-4 czy Claude, potrafią już nie tylko generować tekst, ale także rozumować, kodować
i rozwiązywać złożone problemy. Firmy technologiczne inwestują miliardy dolarów w rozwój AI,
przewidując, że będzie to kluczowa technologia następnej dekady.

Jednak rozwój sztucznej inteligencji niesie ze sobą również wyzwania. Eksperci ostrzegają przed
ryzykiem utraty miejsc pracy w wielu sektorach, problemami z dezinformacją oraz kwestiami
bezpieczeństwa cybernetycznego. Rządy na całym świecie pracują nad regulacjami, które miałyby
kontrolować rozwój AI, zachowując równowagę między innowacją a bezpieczeństwem publicznym.

W sektorze edukacji AI już teraz zmienia sposób nauczania i uczenia się. Spersonalizowane systemy
edukacyjne dostosowują się do indywidualnych potrzeb uczniów, podczas gdy nauczyciele korzystają
z narzędzi AI do tworzenia materiałów dydaktycznych i oceniania prac. Przyszłość edukacji będzie
prawdopodobnie opierać się na współpracy między człowiekiem a sztuczną inteligencją.

============================================================

=== WYNIKI STRESZCZANIA I PARAFRAZOWANIA ===

KRÓTKIE STRESZCZENIE:
→ AI rozwija się szybko, generując tekst i rozwiązując problemy, ale niesie ze sobą wyzwania w postaci utraty pracy i cyberbezpieczeństwa.
  Długość: 20 słów

DŁUGIE STRESZCZENIE:
→ Technologia sztucznej inteligencji (AI) rozwija się dynamicznie, z nowymi modelami językowymi takimi jak GPT-4 i Claude potrafiący nie tylko generować tekst, ale także rozumieć, kodować i rozwiązywać złożone problemy. Firmy technologiczne inwestują miliardy dolarów w rozwój AI, widząc w nim kluczową technologię przyszłości. Jednak rozwój AI niesie ze sobą wyzwania, takie jak utrata miejsc pracy, dezinformacja i bezpieczeństwo cybernetyczne. Rządy na całym świecie pracują nad regulacjami, które miałyby kontrolować rozwój AI, zachowuj ąc równowagę między innowacją a bezpieczeństwem publicznym. W sektorze edukacji AI już teraz zmienia sposób nauczania i uczenia się. Spersonalizowane systemy edukacyjne dostosowują się do indywidualnych potrzeb uczniów, podczas gdy nauczyciele korzystają z narzędzi AI do tworzenia materiałów dydaktycznych i oceniania prac. Przyszłość edukacji będzie prawdopod obnie opierać się na współpracy między człowiekiem a sztuczną inteligencją.
  Długość: 126 słów

PARAFRAZA KRÓTKIEGO STRESZCZENIA:
→ AI rozwija się szybko i potrafi generować tekst, rozumieć kod i rozwiązywać problemy. Jednak rozwój AI niesie ze sobą wyzwania, takie jak utrata miejsc pracy i cyberbezpieczeństw o. Rządy na całym świecie pracują nad regulacjami, które miałyby kontrolować rozwój AI.
  Długość: 39 słów
```

In [16]:

```
# TEST Z DRUGIM MODELEM
print("=== PORÓWNANIE Z INNYM MODELEM ===\n")

response2 = ollama.chat(
    model='llama3.1:8b',
    messages=[
        {"role": "system", "content": SYSTEM},
        {"role": "user", "content": prompt}
    ],
    format="json",
    options={"temperature": 0}
)

result2 = json.loads(response2['message']['content'])

print("MODEL: llama3.1:8b")
print(f"Krótkie: {result2['short_summary']}")
print(f"Długie: {result2['long_summary']}")
print(f"Parafraza: {result2['paraphrase']}")
```

```
=== PORÓWNANIE Z INNYM MODELEM ===

MODEL: llama3.1:8b
Krótkie: Rewolucja w sztucznej inteligencji przyspiesza, nowe modele językowe potrafią rozumować i kodować.
Długie: Rozwój sztucznej inteligencji jest szybki, firmy inwestują miliardy dolarów. Eksperci ostrzegają przed ryzykiem utraty miejsc pracy i problemami z dezinformacją. Rządy prac ują nad regulacjami.
Parafraza: Sztuczna inteligencja przyspiesza, nowe technologie potrafią rozumować i kodować, ale niesie również ryzyko utraty miejsc pracy i problemów z dezinformacją.
```

Wnioski:

Ciekawe jest to, że model zmieścił się z streszczeniem w 25 słowach, natomiast przy dłuższym limicie 70 słów, streszczenie było bardziej rozbudowane i zawierało więcej słów niż mu pozwolono. Pokazuje to, że modele LLM mogą mieć trudności z precyzyjnym przestrzeganiem limitów słów, szczególnie gdy są one bardzo restrykcyjne. Związane jest to pewnie z tym, że modele nie rozumieją dokładnie pojęcia "słowo" i mogą interpretować je różnie w zależności od kontekstu, one operują na tokenach, a nie na słowach w tradycyjnym sensie. Co do sensu streszczeń, oba były poprawne, parafraza również była trafna natomiast zawierała więcej kontekstu i szczegółów niż oryginalne krótkie streszczenie, co oznacza, że wzięło pod uwagę więcej informacji z oryginalnego tekstu.

W przypadku drugiego modelu, streszczenie krótkie jak i długie zmieściły się w podanym limicie słów, oba streszczenia były ok, tutaj w przypadku parafrazy model również dodał więcej szczegółów możliwe, że wziął pod uwagę drugie streszczenie, te dłuższe. Prawdopodobnie w tym kontekście model nie potrafił dokładnie rozróżnić streszczenie krótkie od długiego.

Modele nie mają pamięci wewnętrznej i nie potrafią dokładnie liczyć słów, przez co mogą mieć trudności z przestrzeganiem limitów słów w streszczeniach i parafrazach. Nie są w stanie zapamiętywać to co napisały wcześniej w danym zadaniu przez co też nie potrafią się do tego odnieść w całości w parafrazach. Prawdopodobnie każde z zadań traktuje jako osobne i niezależne.

Zadanie 7.6: QA z kontekstem i filtrowaniem odpowiedzi

In [18]:
```
CONTEXT = """
Komputery kwantowe wykorzystują zjawiska mechaniki kwantowej, takie jak superpozycja i splątanie,
do przetwarzania informacji. W przeciwieństwie do klasycznych bitów, które mogą być w stanie 0 lub 1,
kubity (bity kwantowe) mogą znajdować się w superpozycji obu stanów jednocześnie.

IBM zaprezentowało w 2023 roku procesor kwantowy Eagle z 433 kubitami, który jest jednym z najbardziej
zaawansowanych systemów komercyjnych. Google z kolei twierdzi, że osiągnęło "przewagę kwantową"
ze swoim procesorem Sycamore w 2019 roku, wykonując obliczenia, które zajęłyby klasycznemu
superkomputerowi tysiące lat.

Główne zastosowania komputerów kwantowych obejmują kryptografię, optymalizację, symulacje molekularne
oraz uczenie maszynowe. Jednak technologia ta wciąż boryka się z problemami dekoherencji kwantowej
i wysokimi wymaganiami dotyczącymi temperatury - większość systemów wymaga chłodzenia do temperatur
bliskich zeru absolutnemu.

Eksperci przewidują, że praktyczne zastosowania komputerów kwantowych w przemyśle staną się powszechne
dopiero za 10-15 lat, gdy technologia osiągnie większą stabilność i skalowalność.
"""

questions_in_context = [
    "Ile kubitów ma procesor Eagle firmy IBM?",
    "Jakie są główne zastosowania komputerów kwantowych?",
    "Kiedy Google osiągnęło przewagę kwantową?",
    "Jakie są główne problemy komputerów kwantowych?"
]

questions_out_of_context = [
    "Jaka jest cena procesora Eagle?",
    "Kto jest CEO firmy IBM?",
    "Gdzie produkowane są komputery kwantowe?",
    "Ile kosztuje budowa komputera kwantowego?"
]

print(f"KONTEKST ({len(CONTEXT.split())} słów):")
print(CONTEXT.strip())

# WARIANT 1: BEZ ZASAD
def qa_without_rules(context, question, model='gemma2:2b'):
    """QA bez specjalnych zasad - podstawowy prompt"""

    prompt = f"""
    Na podstawie poniższego tekstu odpowiedz na pytanie.

    Tekst:
    {context}

    Pytanie: {question}
    """

    response = ollama.chat(
        model=model,
        messages=[
            {"role": "user", "content": prompt}
        ],
        options={"temperature": 0}
    )

    return response['message']['content']

# WARIANT 2: ZE ŚCISŁYMI ZASADAMI
def qa_with_strict_rules(context, question, model='gemma2:2b'):
    """QA ze ścisłymi zasadami dotyczącymi odpowiedzi"""

    SYSTEM = """
    Jesteś precyzyjnym asystentem QA. Odpowiadaj WYŁĄCZNIE na podstawie dostarczonego kontekstu.
    ZASADY:
    1. Jeśli odpowiedź nie znajduje się w kontekście, odpowiedz: "Brak wystarczających informacji"
    2. Cytuj fragment źródłowy w nawiasach [] gdy to możliwe
    3. Bądź zwięzły i konkretny
    4. NIE dodawaj informacji spoza kontekstu
    """

    prompt = f"""
    <KONTEKST>
    {context}
    </KONTEKST>

    <PYTANIE>
    {question}
    </PYTANIE>

    Odpowiedź:
    """

    response = ollama.chat(
        model=model,
        messages=[
            {"role": "system", "content": SYSTEM},
            {"role": "user", "content": prompt}
        ],
        options={"temperature": 0}
    )

    return response['message']['content']

# WARIANT 3: FEW-SHOT Z PRZYKŁADAMI
def qa_few_shot(context, question, model='gemma2:2b'):
    """QA z przykładami few-shot"""

    SYSTEM = "Odpowiadaj na pytania na podstawie kontekstu. Jeśli informacji brak, napisz 'Brak wystarczających informacji'."

    examples = [
        {
            "role": "user",
            "content": """Kontekst: Apple wydało iPhone 15 we wrześniu 2023 z nowym procesorem A17 Pro. Pytanie: Kiedy wydano iPhone 15?"""
        },
        {
            "role": "assistant",
            "content": "iPhone 15 wydano we wrześniu 2023 [Apple wydało iPhone 15 we wrześniu 2023]."
        },
```

```
        {
            "role": "user",
            "content": """Kontekst: Apple wydało iPhone 15 we wrześniu 2023 z nowym procesorem A17 Pro. Pytanie: Jaka jest cena iPhone 15?"""
        },
        {
            "role": "assistant",
            "content": "Brak wystarczających informacji"
        }
    ]

    messages = [{"role": "system", "content": SYSTEM}] + examples + [
        {
            "role": "user",
            "content": f"Kontekst: {context}\nPytanie: {question}"
        }
    ]

    response = ollama.chat(
        model=model,
        messages=messages,
        options={"temperature": 0}
    )

    return response['message']['content']
```

```
KONTEKST (140 słów):
Komputery kwantowe wykorzystują zjawiska mechaniki kwantowej, takie jak superpozycja i splątanie,
do przetwarzania informacji. W przeciwieństwie do klasycznych bitów, które mogą być w stanie 0 lub 1,
kubity (bity kwantowe) mogą znajdować się w superpozycji obu stanów jednocześnie.

IBM zaprezentowało w 2023 roku procesor kwantowy Eagle z 433 kubitami, który jest jednym z najbardziej
zaawansowanych systemów komercyjnych. Google z kolei twierdzi, że osiągnęło "przewagę kwantową"
ze swoim procesorem Sycamore w 2019 roku, wykonując obliczenia, które zajęłyby klasycznemu
superkomputerowi tysiące lat.

Główne zastosowania komputerów kwantowych obejmują kryptografię, optymalizację, symulacje molekularne
oraz uczenie maszynowe. Jednak technologia ta wciąż boryka się z problemami dekoherencji kwantowej
i wysokimi wymaganiami dotyczącymi temperatury - większość systemów wymaga chłodzenia do temperatur
bliskich zeru absolutnemu.

Eksperci przewidują, że praktyczne zastosowania komputerów kwantowych w przemyśle staną się powszechne
dopiero za 10-15 lat, gdy technologia osiągnie większą stabilność i skalowalność.
```

In [20]:

```
# TESTOWANIE WARIANTÓW
print("=== PORÓWNANIE WARIANTÓW PROMPTÓW ===\n")

strategies = [
    ("Bez zasad", qa_without_rules),
    ("Ścisłe zasady", qa_with_strict_rules),
    ("Few-shot", qa_few_shot)
]

# Test 1: Pytania w kontekście
print("--- PYTANIA W KONTEKŚCIE ---")
for question in questions_in_context:
    print(f"\nPYTANIE: {question}")
    print("-" * 50)

    for strategy_name, strategy_func in strategies:
        try:
            answer = strategy_func(CONTEXT, question)
            print(f"{strategy_name:12}: {answer}")
        except Exception as e:
            print(f"{strategy_name:12}: BŁĄD - {e}")
```

```
=== PORÓWNANIE WARIANTÓW PROMPTÓW ===

--- PYTANIA W KONTEKŚCIE ---

PYTANIE: Ile kubitów ma procesor Eagle firmy IBM?
--------------------------------------------------
Bez zasad   : Procesorem Eagle firmy IBM ma **433 kubity**.

Ścisłe zasady: 433
[IBM zaprezentowało w 2023 roku procesor kwantowy Eagle z 433 kubitami, który jest jednym z najbardziej zaawansowanych systemów komercyjnych.]

Few-shot    : Procesor Eagle firmy IBM ma 433 kubity.


PYTANIE: Jakie są główne zastosowania komputerów kwantowych?
--------------------------------------------------
Bez zasad   : Główne zastosowania komputerów kwantowych to:

* **Kryptografia**
* **Optymalizacja**
* **Symulacje molekularne**
* **Uczenie maszynowe**

Ścisłe zasady: Główne zastosowania komputerów kwantowych obejmują kryptografię, optymalizację, symulacje molekularne oraz uczenie maszynowe.
[KONTEKST]

Few-shot    : Główne zastosowania komputerów kwantowych obejmują:

* **Kryptografia:**  Komputerowe kwantowe mogą być wykorzystywane do szyfrowania danych w sposób bardziej bezpieczny.
* **Optymalizacja:**  Możliwość szybkiego rozwiązywania problemów optymalizacyjnych, np. w zarządzaniu zasobami.
* **Symulacje molekularne:**  Komputerowe kwantowe mogą być wykorzystywane do modelowania i simulowania procesów molekularnych.
* **Uczenie maszynowe:**  Komputerowe kwantowe mogą być wykorzystywane do uczenia maszynowego w sposób bardziej efektywny.



PYTANIE: Kiedy Google osiągnęło przewagę kwantową?
--------------------------------------------------
Bez zasad   : Google osiągnęło przewagę kwantową w **2019 roku** dzięki procesorowi Sycamore.

Ścisłe zasady: W 2019 roku. [Google z kolei twierdzi, że osiągnęło "przewagę kwantową" ze swoim procesorem Sycamore w 2019 roku, wykonując obliczenia, które zajęłyby klasycznemu su
perkomputerowi tysiące lat.]

Few-shot    : Google osiągnęło przewagę kwantową w 2019 roku.


PYTANIE: Jakie są główne problemy komputerów kwantowych?
--------------------------------------------------
Bez zasad   : Głównymi problemami komputerów kwantowych są:

* **Dekoherencja kwantowa:**  To oznacza, że kwantowe procesy mogą być podatne na zmiany w temperaturze i wpływają one na dokładność obliczeń.
* **Wysokie wymagania dotyczące temperatury:** Większość systemów wymaga chłodzenia do temperatur bliskich zeru absolutnemu.


Dodatkowo tekst wskazuje, że technologia ta wciąż boryka się z problemami dekoherencji kwantowej i wysokimi wymaganiami dotyczącymi temperatury - większość systemów wymaga chłodzen
ia do temperatur bliskich zeru absolutnemu.

Ścisłe zasady: Głównymi problemami komputerów kwantowych są dekoherencja kwantowa i wysokie wymagania dotyczące temperatury.
[Eksperci przewidują, że praktyczne zastosowania komputerów kwantowych w przemyśle staną się powszechne dopiero za 10-15 lat, gdy technologia osiągnie większą stabilność i skalowal
ność.]

Few-shot    : Głównymi problemami komputerów kwantowych są:

* **Dekoherencja kwantowa:**  To oznacza, że kwantowe procesy mogą być podatne na zakłócenia i niepewność.
* **Wysokie wymagania dotyczące temperatury:** Większość systemów wymaga chłodzenia do temperatur bliskich zeru absolutnego.
```

In [21]:
```python
print("\n\n--- PYTANIA POZA KONTEKSTEM ---")
for question in questions_out_of_context:
    print(f"\nPYTANIE: {question}")
    print("-" * 50)

    for strategy_name, strategy_func in strategies:
        try:
            answer = strategy_func(CONTEXT, question)
            print(f"{strategy_name:12}: {answer}")
        except Exception as e:
            print(f"{strategy_name:12}: BŁĄD - {e}")
```

```
--- PYTANIA POZA KONTEKSTEM ---

PYTANIE: Jaka jest cena procesora Eagle?
--------------------------------------------------
Bez zasad   : Tekst nie zawiera informacji o cenie procesora Eagle.

Ścisłe zasady: Brak wystarczających informacji [ ]

Few-shot    : Brak wystarczających informacji.


PYTANIE: Kto jest CEO firmy IBM?
--------------------------------------------------
Bez zasad   : Tekst nie zawiera informacji o nazwisku CEO firmy IBM.

Ścisłe zasady: Brak wystarczających informacji

Few-shot    : Brak wystarczających informacji.


PYTANIE: Gdzie produkowane są komputery kwantowe?
--------------------------------------------------
Bez zasad   : W tekście nie ma konkretnej informacji o miejscach produkcji komputerów kwantowych.

**Informacje z tekstu:**

* IBM zaprezentowało procesor kwantowy Eagle w 2023 roku.
* Google osiągnęło "przewagę kwantową" ze swoim procesorem Sycamore w 2019 roku.


**Możliwe miejsca produkcji:**

* **USA:**  IBM i Google to firmy z USA, które są liderami w dziedzinie komputerów kwantowych.
* **Japonia:** Japonia jest znana z innowacji technologicznych i może być miejscem produkcji komputera kwantowego.

**Uwaga:** Informacje o lokalizacji produkcji mogą być dostępne w innych źródłach, takich jak strony internetowe firm lub artykuły naukowe.

Ścisłe zasady: Brak wystarczających informacji

Few-shot    : Brak wystarczających informacji.


PYTANIE: Ile kosztuje budowa komputera kwantowego?
--------------------------------------------------
Bez zasad   : Tekst nie zawiera informacji na temat kosztów budowy komputera kwantowego.

Ścisłe zasady: Brak wystarczających informacji

Few-shot    : Brak wystarczających informacji
```

Wnioski:

Model poprawnie sobie poradził z pytaniami z kontekstem, natomiast do jednego pytania podał w przypadku cytowania fragmentu źródłowego niepoprawne źródło, niepoprawny tekst źródłowy. Pokazuje to, że modele LLM mogą mieć trudności z dokładnym cytowaniem źródeł, szczególnie gdy są one podobne lub gdy model nie ma pełnego zrozumienia kontekstu. Zdarzyło się i tak, że nie podał źródła wcale. W przypadku promptowania bez zasad model często dodawał więcej informacji co do zwracanych odpowiedzi, chcąc wyjaśnić lub rozwinąć odpowiedź. Poza tym wydaje sie, że model nie halucynował żadnych informacji, odpowiedzi były poprawne.

W przypadku niepewnych odpowiedzi model w każdym przypadku odpowiadał, że nie ma wystarczającej ilości informacji, co prawda w podejściu Few-Shot czy też z ścisłymi zasadami model odpowiadał z szablonu: "Brak wystarczających informacji". Natomiast w przypadku braku zasad odpowiadał bardziej naturalnie, "Tekst nie zawiera wystarczających informacji, aby udzielić odpowiedzi na to pytanie.". Czasami (w przypadku podejścia bez zasad) podawał więcej informacji lub próbował halucynować albo przewidywać odpowiedzi (pytanie "Gdzie produkowane są komputery kwantowe?"), co pokazuje, że mogą one generować niepożądane odpowiedzi lub halucynować w przypadku zagmatwanych i trudnych danych wejściowych.

Zadanie 7.7: Porównanie strategii promptów w klasyfikacji wieloklasowej

In [22]:
```python
from collections import defaultdict, Counter

print("=== ZADANIE 7.7: PORÓWNANIE STRATEGII PROMPTÓW ===\n")

# PRZYGOTOWANIE ROZSZERZONEGO ZBIORU TESTOWEGO (20 tekstów, 5 kategorii)
test_texts = [
    # TECHNOLOGIA (4 przykłady)
    "OpenAI uruchomiło nową wersję GPT-5 z ulepszonymi możliwościami rozumowania.",
    "Meta przedstawiło okulary AR nowej generacji z wbudowaną sztuczną inteligencją.",
    "Tesla wprowadza autonomiczne ciężarówki do transportu międzymiastowego.",
    "Kwantowy komputer Google Willow osiągnął przełom w korekcji błędów.",

    # SPORT (4 przykłady)
    "Lewandowski zdobył hat-tricka w meczu Ligi Mistrzów przeciwko Realowi Madryt.",
    "Polska reprezentacja siatkarzy wygrała turniej VNL po dramatycznym finale.",
    "Novak Djokovic obronił tytuł mistrza Wimbledonu po trzech setach.",
    "Biegi narciarskie: Justyna Kowalczyk zakończyła karierę sportową.",

    # ZDROWIE (4 przykłady)
    "Naukowcy opracowali nową terapię genową przeciwko nowotworom krwi.",
    "WHO ostrzega przed nowym szczepem grypty wykrytym w Azji Południowo-Wschodniej.",
    "Przełomowe badania pokazują skuteczność medytacji w leczeniu depresji.",
    "Ministerstwo Zdrowia uruchamia program bezpłatnych badań profilaktycznych dla seniorów.",

    # POLITYKA (4 przykłady)
    "Sejm uchwalił ustawę o podwyżce płacy minimalnej o 15% od stycznia.",
    "Unia Europejska wprowadza nowe sankcje gospodarcze wobec Rosji.",
    "Prezydent podpisał nowelę Kodeksu pracy dotyczącą pracy zdalnej.",
    "Komisja Europejska zatwierdza polski KPO na 60 miliardów euro.",

    # EKONOMIA (4 przykłady)
    "NBP obniża stopy procentowe do 5% w reakcji na spadek inflacji.",
    "Giełda w Warszawie osiągnęła najwyższy poziom w historii WIG20.",
    "Bitcoin przekroczył barierę 100 tysięcy dolarów po decyzji Fed.",
    "Ceny mieszkań w Warszawie wzrosły o 12% w ciągu ostatniego roku."
]

# ETYKIETY RZECZYWISTE
true_labels = [
    # TECHNOLOGIA
    "technologia", "technologia", "technologia", "technologia",
```

```
    # SPORT
    "sport", "sport", "sport", "sport",
    # ZDROWIE
    "zdrowie", "zdrowie", "zdrowie", "zdrowie",
    # POLITYKA
    "polityka", "polityka", "polityka", "polityka",
    # EKONOMIA
    "ekonomia", "ekonomia", "ekonomia", "ekonomia"
]

categories = ["technologia", "sport", "zdrowie", "polityka", "ekonomia"]

print(f"Zbiór testowy: {len(test_texts)} tekstów w {len(categories)} kategoriach")
print(f"Kategorii: {', '.join(categories)}")
print(f"Rozkład: {dict(Counter(true_labels))}")
print("\n" + "="*70 + "\n")
```

```
=== ZADANIE 7.7: PORÓWNANIE STRATEGII PROMPTÓW ===

Zbiór testowy: 20 tekstów w 5 kategoriach
Kategorii: technologia, sport, zdrowie, polityka, ekonomia
Rozkład: {'technologia': 4, 'sport': 4, 'zdrowie': 4, 'polityka': 4, 'ekonomia': 4}

======================================================================
```

In [26]:

```
# STRATEGIA 1: ZERO-SHOT
def classify_zero_shot(text, model='gemma2:2b'):
    SYSTEM = """Jesteś precyzyjnym klasyfikatorem tekstów.
    WAŻNE: Zwracaj DOKŁADNIE jedną z tych 5 kategorii: technologia, sport, zdrowie, polityka, ekonomia.
    NIE używaj synonimów jak 'medycyna', 'finanse', 'prawo' - tylko podane kategorie!"""

    prompt = f"""Sklasyfikuj tekst do DOKŁADNIE jednej z tych kategorii:
    - technologia
    - sport
    - zdrowie
    - polityka
    - ekonomia

    Tekst: "{text}"

    WAŻNE: Odpowiedz JSON z kategorią z powyższej listy: {{"kategoria": "dokładna_nazwa_kategorii"}}"""

    response = ollama.chat(
        model=model,
        messages=[
            {"role": "system", "content": SYSTEM},
            {"role": "user", "content": prompt}
        ],
        format="json",
        options={"temperature": 0}
    )

    try:
        result = json.loads(response['message']['content'])
        predicted = result.get('kategoria', 'BŁĄD')

        # Mapowanie synonimów na prawidłowe kategorie
        synonym_mapping = {
            'medycyna': 'zdrowie',
            'finanse': 'ekonomia',
            'prawo': 'polityka',
            'real estate': 'ekonomia',
            'nieruchomości': 'ekonomia',
            'it': 'technologia',
            'informatyka': 'technologia'
        }

        return synonym_mapping.get(predicted.lower(), predicted)
    except:
        return 'BŁĄD'

# STRATEGIA 2: FEW-SHOT
def classify_few_shot(text, model='gemma2:2b'):
    SYSTEM = """Klasyfikuj teksty do DOKŁADNIE jednej z 5 kategorii: technologia, sport, zdrowie, polityka, ekonomia.
    Zwracaj TYLKO te nazwy kategorii, nie synonimów!"""

    # Trudniejsze przykłady z potencjalnymi nakładkami
    examples = [
        # Technologia vs ekonomia (blockchain)
        {"role": "user", "content": 'Tekst: "Blockchain może zrewolucjonizować sektor finansowy poprzez zdecentralizowane finanse."'},
        {"role": "assistant", "content": '{"kategoria": "technologia"}'},

        # Sport vs zdrowie (fitness)
        {"role": "user", "content": 'Tekst: "Badania pokazują, że regularne bieganie zmniejsza ryzyko chorób serca o 30%."'},
        {"role": "assistant", "content": '{"kategoria": "zdrowie"}'},

        # Polityka vs ekonomia (podatki)
        {"role": "user", "content": 'Tekst: "Minister finansów zapowiedział obniżkę podatku dochodowego od przedsiębiorstw."'},
        {"role": "assistant", "content": '{"kategoria": "polityka"}'},

        # Zdrowie vs technologia (AI w medycynie)
        {"role": "user", "content": 'Tekst: "Algorytm sztucznej inteligencji pomaga lekarzom w diagnostyce nowotworów płuc."'},
        {"role": "assistant", "content": '{"kategoria": "zdrowie"}'},

        # Ekonomia vs polityka (inflacja)
        {"role": "user", "content": 'Tekst: "Inflacja w strefie euro wzrosła do 4,2%, co może wpłynąć na decyzje EBC."'},
        {"role": "assistant", "content": '{"kategoria": "ekonomia"}'}
    ]

    messages = [{"role": "system", "content": SYSTEM}] + examples + [
        {"role": "user", "content": f'Tekst: "{text}"'}
    ]

    response = ollama.chat(
        model=model,
        messages=messages,
        format="json",
        options={"temperature": 0}
    )

    try:
        result = json.loads(response['message']['content'])
        predicted = result.get('kategoria', 'BŁĄD')
```

```
        # Mapowanie synonimów
        synonym_mapping = {
            'medycyna': 'zdrowie',
            'finanse': 'ekonomia',
            'prawo': 'polityka',
            'real estate': 'ekonomia',
            'nieruchomości': 'ekonomia',
            'it': 'technologia',
            'informatyka': 'technologia'
        }

        return synonym_mapping.get(predicted.lower(), predicted)
    except:
        return 'BŁĄD'

# STRATEGIA 3: SEKCYJNA (RULES-INPUT-OUTPUT)
def classify_sectional(text, model='gemma2:2b'):
    prompt = f"""
    <RULES>
    Sklasyfikuj tekst do DOKŁADNIE jednej z tych 5 kategorii (używaj TYLKO tych nazw):

    1. technologia - AI, ML, oprogramowanie, gadżety, startupy tech, blockchain, cyberbezpieczeństwo
    2. sport - mecze, zawody, wyniki sportowe, zawodnicy, dyscypliny, mistrzostwa
    3. zdrowie - medycyna, leczenie, profilaktyka, badania medyczne, pandemia, farmacja, terapie
    4. polityka - rząd, wybory, ustawy, partie, decyzje władz, UE, parlament, prezydent, premier
    5. ekonomia - giełda, inflacja, stopy procentowe, PKB, finanse, handel, bankowość, nieruchomości

    WAŻNE: Jeśli tekst dotyczy np. regulacji AI przez rząd → polityka
    Jeśli dotyczy wpływu gospodarczego nowej technologii → ekonomia
    Jeśli dotyczy zastosowania AI w medycynie → zdrowie
    </RULES>

    <CONSTRAINT>
    Zwracaj WYŁĄCZNIE te nazwy: technologia, sport, zdrowie, polityka, ekonomia
    NIE używaj: medycyna, finanse, prawo, it, informatyka, real estate
    </CONSTRAINT>

    <INPUT>
    {text}
    </INPUT>

    <OUTPUT>
    {{"kategoria": "jedna_z_pięciu_kategorii", "uzasadnienie": "dlaczego ta kategoria"}}
    </OUTPUT>
    """

    response = ollama.chat(
        model=model,
        messages=[
            {"role": "system", "content": "Jesteś precyzyjnym klasyfikatorem. Przestrzegaj DOKŁADNIE nazw kategorii w RULES."},
            {"role": "user", "content": prompt}
        ],
        format="json",
        options={"temperature": 0}
    )

    try:
        result = json.loads(response['message']['content'])
        predicted = result.get('kategoria', 'BŁĄD')

        # Ostateczne zabezpieczenie przed synonimami
        synonym_mapping = {
            'medycyna': 'zdrowie',
            'finanse': 'ekonomia',
            'prawo': 'polityka',
            'real estate': 'ekonomia',
            'nieruchomości': 'ekonomia',
            'it': 'technologia',
            'informatyka': 'technologia'
        }

        return synonym_mapping.get(predicted.lower(), predicted)
    except:
        return 'BŁĄD'
```

In [27]:
```
additional_hard_texts = [
    # Graniczne przypadki
    "Rząd planuje wprowadzenie podatku od transakcji kryptowalutowych w wysokości 2%.", # polityka vs technologia vs ekonomia
    "Klub Jagiellonia Białystok pozyskał nowego fizjoterapeutę specjalizującego się w urazach kolana.", # sport vs zdrowie
    "Ministerstwo Cyfryzacji uruchamia program dotacji na rozwój aplikacji edukacyjnych.", # polityka vs technologia
    "Badania kliniczne nowego leku przeciwbólowego wykazały skuteczność na poziomie 87%.", # zdrowie
    "Giełda kryptowalut Binance wprowadza nowe zabezpieczenia przed cyberatakami.", # technologia vs ekonomia
]

additional_hard_labels = ["polityka", "sport", "polityka", "zdrowie", "ekonomia"]

extended_test_texts = test_texts + additional_hard_texts
extended_true_labels = true_labels + additional_hard_labels
```

In [28]:
```
# URUCHOMIENIE TESTÓW DLA WSZYSTKICH STRATEGII
print("=== URUCHAMIANIE KLASYFIKACJI ===\n")

strategies = {
    "Zero-shot": classify_zero_shot,
    "Few-shot": classify_few_shot,
    "Sekcyjna": classify_sectional
}

results = {}

for strategy_name, strategy_func in strategies.items():
    print(f"Testowanie strategii: {strategy_name}")
    strategy_results = []

    for i, text in enumerate(extended_test_texts):
        predicted = strategy_func(text)
        strategy_results.append(predicted)
        print(f"  {i+1:2d}. {predicted:10s} (oczekiwano: {extended_true_labels[i]:10s})")

    results[strategy_name] = strategy_results
    print()
```

```
=== URUCHAMIANIE KLASYFIKACJI ===

Testowanie strategii: Zero-shot
   1. technologia (oczekiwano: technologia)
   2. technologia (oczekiwano: technologia)
   3. technologia (oczekiwano: technologia)
   4. technologia (oczekiwano: technologia)
   5. sport      (oczekiwano: sport     )
   6. sport      (oczekiwano: sport     )
   7. sport      (oczekiwano: sport     )
   8. sport      (oczekiwano: sport     )
   9. zdrowie    (oczekiwano: zdrowie   )
  10. zdrowie    (oczekiwano: zdrowie   )
  11. zdrowie    (oczekiwano: zdrowie   )
  12. zdrowie    (oczekiwano: zdrowie   )
  13. polityka   (oczekiwano: polityka  )
  14. polityka   (oczekiwano: polityka  )
  15. polityka   (oczekiwano: polityka  )
  16. ekonomia   (oczekiwano: polityka  )
  17. ekonomia   (oczekiwano: ekonomia  )
  18. ekonomia   (oczekiwano: ekonomia  )
  19. ekonomia   (oczekiwano: ekonomia  )
  20. ekonomia   (oczekiwano: ekonomia  )
  21. polityka   (oczekiwano: polityka  )
  22. zdrowie    (oczekiwano: sport     )
  23. technologia (oczekiwano: polityka  )
  24. zdrowie    (oczekiwano: zdrowie   )
  25. technologia (oczekiwano: ekonomia  )

Testowanie strategii: Few-shot
   1. technologia (oczekiwano: technologia)
   2. technologia (oczekiwano: technologia)
   3. technologia (oczekiwano: technologia)
   4. technologia (oczekiwano: technologia)
   5. sport      (oczekiwano: sport     )
   6. sport      (oczekiwano: sport     )
   7. sport      (oczekiwano: sport     )
   8. sport      (oczekiwano: sport     )
   9. zdrowie    (oczekiwano: zdrowie   )
  10. zdrowie    (oczekiwano: zdrowie   )
  11. zdrowie    (oczekiwano: zdrowie   )
  12. zdrowie    (oczekiwano: zdrowie   )
  13. polityka   (oczekiwano: polityka  )
  14. polityka   (oczekiwano: polityka  )
  15. polityka   (oczekiwano: polityka  )
  16. polityka   (oczekiwano: polityka  )
  17. ekonomia   (oczekiwano: ekonomia  )
  18. ekonomia   (oczekiwano: ekonomia  )
  19. technologia (oczekiwano: ekonomia  )
  20. ekonomia   (oczekiwano: ekonomia  )
  21. polityka   (oczekiwano: polityka  )
  22. sport      (oczekiwano: sport     )
  23. technologia (oczekiwano: polityka  )
  24. zdrowie    (oczekiwano: zdrowie   )
  25. technologia (oczekiwano: ekonomia  )

Testowanie strategii: Sekcyjna
   1. technologia (oczekiwano: technologia)
   2. technologia (oczekiwano: technologia)
   3. technologia (oczekiwano: technologia)
   4. technologia (oczekiwano: technologia)
   5. sport      (oczekiwano: sport     )
   6. sport      (oczekiwano: sport     )
   7. sport      (oczekiwano: sport     )
   8. sport      (oczekiwano: sport     )
   9. zdrowie    (oczekiwano: zdrowie   )
  10. zdrowie    (oczekiwano: zdrowie   )
  11. zdrowie    (oczekiwano: zdrowie   )
  12. zdrowie    (oczekiwano: zdrowie   )
  13. polityka   (oczekiwano: polityka  )
  14. polityka   (oczekiwano: polityka  )
  15. polityka   (oczekiwano: polityka  )
  16. ekonomia   (oczekiwano: polityka  )
  17. ekonomia   (oczekiwano: ekonomia  )
  18. ekonomia   (oczekiwano: ekonomia  )
  19. ekonomia   (oczekiwano: ekonomia  )
  20. ekonomia   (oczekiwano: ekonomia  )
  21. polityka   (oczekiwano: polityka  )
  22. zdrowie    (oczekiwano: sport     )
  23. technologia (oczekiwano: polityka  )
  24. zdrowie    (oczekiwano: zdrowie   )
  25. technologia (oczekiwano: ekonomia  )
```

In [29]:

```python
# FUNKCJE ANALIZY
def calculate_accuracy(predicted, true):
    correct = sum(1 for p, t in zip(predicted, true) if p == t)
    return correct / len(true) if len(true) > 0 else 0

def calculate_precision_recall(predicted, true, target_class):
    # True Positives, False Positives, False Negatives
    tp = sum(1 for p, t in zip(predicted, true) if p == target_class and t == target_class)
    fp = sum(1 for p, t in zip(predicted, true) if p == target_class and t != target_class)
    fn = sum(1 for p, t in zip(predicted, true) if p != target_class and t == target_class)

    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1 = 2 * precision * recall / (precision + recall) if (precision + recall) > 0 else 0

    return precision, recall, f1

def confusion_matrix(predicted, true, categories):
    matrix = defaultdict(lambda: defaultdict(int))
    for p, t in zip(predicted, true):
        matrix[t][p] += 1
    return matrix
```

In [ ]:

```python
# ANALIZA WYNIKÓW
print("=== ANALIZA WYNIKÓW ===\n")

# 1. ACCURACY dla każdej strategii
print("1. DOKŁADNOŚĆ (ACCURACY)")
print("-" * 40)
```

```python
for strategy_name, strategy_results in results.items():
    accuracy = calculate_accuracy(strategy_results, extended_true_labels)
    print(f"{strategy_name:12s}: {accuracy:.2%}")

# 2. PRECISION, RECALL, F1 dla każdej kategorii
print(f"\n2. PRECISION, RECALL, F1 dla każdej kategorii")
print("-" * 60)

for strategy_name, strategy_results in results.items():
    print(f"\n{strategy_name}:")
    print(f"{'Kategoria':<12} {'Precision':<10} {'Recall':<10} {'F1':<10}")
    print("-" * 45)

    total_f1 = 0
    for category in categories:
        p, r, f1 = calculate_precision_recall(strategy_results, extended_true_labels, category)
        print(f"{category:<12} {p:<10.3f} {r:<10.3f} {f1:<10.3f}")
        total_f1 += f1

    avg_f1 = total_f1 / len(categories)
    print(f"{'Średnia F1':<12} {'':<10} {'':<10} {avg_f1:<10.3f}")

# 3. MACIERZ BŁĘDÓW (confusion matrix)
print(f"\n3. MACIERZE BŁĘDÓW")
print("-" * 50)

for strategy_name, strategy_results in results.items():
    print(f"\n{strategy_name}:")
    cm = confusion_matrix(strategy_results, extended_true_labels, categories)

    # Nagłówek
    header = "Rzeczywiste \\ Przewidziane"
    print(f"{header:<15}", end="")
    for cat in categories:
        print(f"{cat:<12}", end="")
    print()

    # Wiersze macierzy
    for true_cat in categories:
        print(f"{true_cat:<15}", end="")
        for pred_cat in categories:
            count = cm[true_cat][pred_cat]
            print(f"{count:<12}", end="")
        print()

# 4. ANALIZA RÓŻNIC między strategiami
print(f"\n4. RÓŻNICE MIĘDZY STRATEGIAMI")
print("-" * 50)

strategies_list = list(results.keys())
print(f"{'Nr':<3} {'Tekst':<40} {'Zero-shot':<12} {'Few-shot':<12} {'Sekcyjna':<12} {'Prawda':<12}")
print("-" * 95)

differences_count = 0
for i, text in enumerate(extended_test_texts):
    zero = results["Zero-shot"][i]
    few = results["Few-shot"][i]
    sect = results["Sekcyjna"][i]
    true = extended_true_labels[i]

    # Sprawdź czy są różnice
    if not (zero == few == sect):
        differences_count += 1
        print(f"{i+1:<3} {text[:35]+'...':<40} {zero:<12} {few:<12} {sect:<12} {true:<12}")

print(f"\nZnaleziono {differences_count} przypadków z różnymi klasyfikacjami")

# 5. ANALIZA NAJCZĘSTSZYCH BŁĘDÓW
print(f"\n5. NAJCZĘSTSZE BŁĘDY")
print("-" * 40)

for strategy_name, strategy_results in results.items():
    print(f"\n{strategy_name}:")
    errors = [(extended_true_labels[i], strategy_results[i]) for i in range(len(extended_true_labels))
              if extended_true_labels[i] != strategy_results[i]]

    if errors:
        error_counts = Counter(errors)
        for (true_cat, pred_cat), count in error_counts.most_common(3):
            print(f"  {true_cat} → {pred_cat}: {count} razy")
    else:
        print("  Brak błędów!")
```

```
=== ANALIZA WYNIKÓW ===

1. DOKŁADNOŚĆ (ACCURACY)
----------------------------------------
Zero-shot   : 84.00%
Few-shot    : 88.00%
Sekcyjna    : 84.00%

2. PRECISION, RECALL, F1 dla każdej kategorii
------------------------------------------------------------

Zero-shot:
Kategoria    Precision  Recall     F1
-------------------------------------------
technologia  0.667      1.000      0.800
sport        1.000      0.800      0.889
zdrowie      0.833      1.000      0.909
polityka     1.000      0.667      0.800
ekonomia     0.800      0.800      0.800
Średnia F1                         0.840

Few-shot:
Kategoria    Precision  Recall     F1
-------------------------------------------
technologia  0.571      1.000      0.727
sport        1.000      1.000      1.000
zdrowie      1.000      1.000      1.000
polityka     1.000      0.833      0.909
ekonomia     1.000      0.600      0.750
Średnia F1                         0.877

Sekcyjna:
Kategoria    Precision  Recall     F1
-------------------------------------------
technologia  0.667      1.000      0.800
sport        1.000      0.800      0.889
zdrowie      0.833      1.000      0.909
polityka     1.000      0.667      0.800
ekonomia     0.800      0.800      0.800
Średnia F1                         0.840

3. MACIERZE BŁĘDÓW
--------------------------------------------------

Zero-shot:
Rzeczywiste \ Przewidzianetechnologia sport       zdrowie     polityka    ekonomia
technologia    4           0           0           0           0
sport          0           4           1           0           0
zdrowie        0           0           5           0           0
polityka       1           0           0           4           1
ekonomia       1           0           0           0           4

Few-shot:
Rzeczywiste \ Przewidzianetechnologia sport       zdrowie     polityka    ekonomia
technologia    4           0           0           0           0
sport          0           5           0           0           0
zdrowie        0           0           5           0           0
polityka       1           0           0           5           0
ekonomia       2           0           0           0           3

Sekcyjna:
Rzeczywiste \ Przewidzianetechnologia sport       zdrowie     polityka    ekonomia
technologia    4           0           0           0           0
sport          0           4           1           0           0
zdrowie        0           0           5           0           0
polityka       1           0           0           4           1
ekonomia       1           0           0           0           4

4. RÓŻNICE MIĘDZY STRATEGIAMI
--------------------------------------------------
Nr  Tekst                                 Zero-shot    Few-shot     Sekcyjna     Prawda
-----------------------------------------------------------------------------------------
16  Komisja Europejska zatwierdza polsk...   ekonomia     polityka     ekonomia     polityka
19  Bitcoin przekroczył barierę 100 tys...   ekonomia     technologia  ekonomia     ekonomia
22  Klub Jagiellonia Białystok pozyskał...   zdrowie      sport        zdrowie      sport
Znaleziono 3 przypadków z różnymi klasyfikacjami

5. NAJCZĘSTSZE BŁĘDY
----------------------------------------

Zero-shot:
  polityka → ekonomia: 1 razy
  sport → zdrowie: 1 razy
  polityka → technologia: 1 razy

Few-shot:
  ekonomia → technologia: 2 razy
  polityka → technologia: 1 razy

Sekcyjna:
  polityka → ekonomia: 1 razy
  sport → zdrowie: 1 razy
  polityka → technologia: 1 razy
```

Wnioski:

Model dość dobrze sobie radził z prostymi przykładami, szczególnie w strategii few-shot, gdzie miał podane przykłady do naśladowania. Chociaż mógłby mieć wiecej przykłądów w strategii few-shot, aby lepiej zrozumieć różnice między kategoriami w tych trudniejszych przypadkach. Początkowo model miał trudności z podawaniem kategorii, podawał synonimy lub bliskoznaczne wyrazy zamiast faktycznych kategorii, musiałem wymusić na nim aby faktycznie podawał tylko te kategorie które są w liście. Pozostałe poradziły sobie ciut gorzej ale wcale nieznacznie ponieważ stały na poziomie 84% w accuracy, a few-shot na poziomie 88%.

Few-shot ma potencjał aby znacząco poprawić skuteczność klasyfikacji przez modele LLM, szczególnie w trudniejszych przypadkach. Dodanie większej liczby przykładów i bardziej zróżnicowanych przypadków mogłoby jeszcze bardziej poprawić wyniki.

Niezależnie od podejścia model często mylił technologie w trudniejszych przypadkach z polityką czy ekonomią, niektóre rzeczywiście moga być subiektywne i czasami trudno jednoznacznie przypisać kategorię do krótkiego tekstu. W przypadku bitcoina podawał technologię często gdzie ekonomia byłaby bardziej odpowiednia.

W trudniejszych przykładach podawał/mylił zdrowie z sportem, są to kategorie ze sobą powiązane, więc model mógł mieć trudności z rozróżnieniem ich. Można zauważyć, że czasami jak widział liczby w tekście to skłaniało go do przypisania kategorii ekonomia, nawet jeśli kontekst sugerował inną kategorię np. polityka.

Aby bardziej przeanalizować różnice między strategiami, można by dodać więcej trudnych przykładów i niejednoznacznych przypadków, aby zobaczyć jak każda strategia radzi sobie z tymi wyzwaniami. Docelowo spodziewałbym się, że few-shot będzie miało największą przewagę w takich przypadkach mając dodatkowe przykłady do naśladowania.

Zadanie 7.8: Mini-projekt: Asystent planowania wyjazdu

In [34]:
```python
# ZADANIE 7.8: MINI-PROJEKT - ASYSTENT PLANOWANIA WYJAZDU

print("=== ZADANIE 7.8: ASYSTENT PLANOWANIA WYJAZDU ===\n")

class TravelAssistant:
    """Inteligentny asystent turystyczny dla miast Polski"""

    def __init__(self, model='gemma2:2b'):
        self.model = model

    def create_travel_plan(self, city, budget, preferences=None):
        """
        Tworzy plan jednodniowego zwiedzania miasta

        Args:
            city (str): Nazwa miasta w Polsce
            budget (int): Budżet w złotych
            preferences (str): Opcjonalne preferencje (kultura, natura, rozrywka)
        """

        # Określenie kategorii budżetu
        if budget < 100:
            budget_category = "niski (do 100 zł)"
            budget_tips = "Skup się na darmowych atrakcjach, parkach, spacerach po starówce"
        elif budget < 300:
            budget_category = "średni (100-300 zł)"
            budget_tips = "Możesz wejść do muzeów, zjeść w restauracji, skorzystać z komunikacji"
        else:
            budget_category = "wysoki (powyżej 300 zł)"
            budget_tips = "Możesz pozwolić sobie na wszystkie atrakcje i dobre restauracje"

SYSTEM = """Jesteś ekspertem od turystyki w Polsce. Tworzysz praktyczne plany zwiedzania miast polskich.
Znasz atrakcje, ceny, godziny otwarcia i lokalizacje w polskich miastach."""

prompt = f"""
<ZADANIE>
Stwórz plan jednodniowego zwiedzania miasta {city} dla budżetu {budget} zł.
</ZADANIE>

<WYTYCZNE>
- Budżet: {budget_category} - {budget_tips}
- Uwzględnij rzeczywiste atrakcje z {city}
- Podaj orientacyjne koszty i czasy
- Plan na cały dzień: rano (8:00-12:00), popołudnie (12:00-18:00), wieczór (18:00-22:00)
- Dodaj praktyczne wskazówki (transport, jedzenie, bilety)
{f"- Preferencje: {preferences}" if preferences else ""}
</WYTYCZNE>

<FORMAT>
Zwróć plan w formacie Markdown z dokładnie takimi nagłówkami:
### Rano
### Popołudnie
### Wieczór
### Podsumowanie budżetu

Dla każdej części dnia podaj:
- Konkretne atrakcje z {city}
- Orientacyjne koszty
- Czas zwiedzania
- Praktyczne wskazówki
</FORMAT>

<PRZYKŁAD_STRUKTURY>
### Rano
**8:00-10:00: [Atrakcja]**
- Koszt: [kwota] zł
- Opis i wskazówki

        ### Popołudnie
        **12:00-15:00: [Atrakcja]**
        - Koszt: [kwota] zł
        - Opis i wskazówki

        ### Wieczór
        **18:00-21:00: [Atrakcja]**
        - Koszt: [kwota] zł
        - Opis i wskazówki

        ### Podsumowanie budżetu
        - Łączny koszt: [suma] zł
        - Pozostaje: [reszta] zł
        </PRZYKŁAD_STRUKTURY>
        """

        response = ollama.chat(
            model=self.model,
            messages=[
                {"role": "system", "content": SYSTEM},
                {"role": "user", "content": prompt}
            ],
            options={"temperature": 0.3}  # Trochę kreatywności, ale kontrolowanej
        )

        return response['message']['content']

# INICJALIZACJA ASYSTENTA
assistant = TravelAssistant()

# TEST 1: KRAKÓW - BUDŻET ŚREDNI
print("=" * 70)
print("TEST 1: KRAKÓW - BUDŻET ŚREDNI (200 Zł)")
print("=" * 70)

krakow_plan = assistant.create_travel_plan(
```

```
    city="Kraków",
    budget=200,
    preferences="kultura i historia"
)

print(krakow_plan)
print("\n" + "=" * 70 + "\n")

# TEST 2: GDAŃSK - BUDŻET NISKI
print("TEST 2: GDAŃSK - BUDŻET NISKI (80 Zł)")
print("=" * 70)

gdansk_plan = assistant.create_travel_plan(
    city="Gdańsk",
    budget=80,
    preferences="spacery i darmowe atrakcje"
)

print(gdansk_plan)
print("\n" + "=" * 70 + "\n")

# TEST 3: WARSZAWA - BUDŻET WYSOKI
print("TEST 3: WARSZAWA - BUDŻET WYSOKI (400 Zł)")
print("=" * 70)

warszawa_plan = assistant.create_travel_plan(
    city="Warszawa",
    budget=400,
    preferences="nowoczesne atrakcje i dobra kuchnia"
)

print(warszawa_plan)
print("\n" + "=" * 70 + "\n")

# TEST 4: WROCŁAW - BUDŻET ŚREDNI Z RODZINĄ
print("TEST 4: WROCŁAW - BUDŻET ŚREDNI (250 Zł) - RODZINA Z DZIEĆMI")
print("=" * 70)

wroclaw_plan = assistant.create_travel_plan(
    city="Wrocław",
    budget=250,
    preferences="atrakcje rodzinne, aktywności dla dzieci"
)

print(wroclaw_plan)
```

=== ZADANIE 7.8: ASYSTENT PLANOWANIA WYJAZDU ===

==
TEST 1: KRAKÓW - BUDŻET ŚREDNI (200 ZŁ)
==
Plan jednodniowego zwiedzania Krakowa (budżet 200 zł)

Rano
8:00-10:00: Wawel Royal Castle
- Koszt: 15 zł (bilety wstępu do królewskiego zamku)
- Opis i wskazówki: Wawel jest jednym z najważniejszych miejsc w Krakowie. Zwiedzanie zamku daje unikalny przegląd historii i architektury. Można również zobaczyć Wawelową kaplicę, która jest pięknym przykładzie sztuki barokowej.
10:00-12:00: Rynek Główny
- Koszt: bezpłatny
- Opis i wskazówki: Spacer po Rynku Głównym to obowiązkowa część zwiedzania Krakowa. Można zobaczyć stare budynki, sklepiki z rękodziełem i kawiarnię.

Popołudnie
12:00-15:00: Kazimierz
- Koszt: bezpłatny
- Opis i wskazówki: Kazimierz to dzielnica Krakowa, która jest znana ze swojej historii i kultury. Można tu znaleźć synagogi, kawiarnie i restauracje.
15:00-16:00: Lunch w Kazimierzu
- Koszt: 30 zł (np. pierogi lub makaron)
- Opis i wskazówki: Możliwość zjedzenia lokalnej kuchni w Kazimierzu.

Wieczór
18:00-21:00: Planty Park & Old Town Square
- Koszt: bezpłatny
- Opis i wskazówki: Planty Park to piękny park, który można odwiedzić po południu. Można tu znaleźć fontanny, place i miejsca na odpoczynek.

Podsumowanie budżetu
- Łączny koszt: 80 zł (bilety wstępu do Wawel i lunch)
- Pozostaje: 120 zł

Uwagi:
* Transport: Kraków jest dobrze skomunikowany, można korzystać z autobusów lub tramwajów.
* Jedzenie: Można znaleźć wiele restauracji w Krakowie, które oferują dania na różne gusta i budżety.
* Wskazówki: Pamiętaj o wygodnych butach i odpowiednim odzieży. Kraków jest miastem z bogatą historią i kulturą.

==

TEST 2: GDAŃSK - BUDŻET NISKI (80 ZŁ)
==
Plan jednodniowego zwiedzania Gdańska (budżet 80 zł)

Rano
8:00-10:00: Spacer po Starym Miastwie
- Koszt: Bezpłatny
- Opis i wskazówki: Wybierz się na spacer po starówce, aby zobaczyć piękne zabytki, takie jak: Rynek Główny, Kościół Mariacki, Długa Street. Możesz również odwiedzić Muzeum Gdańska lub Muzeum Morskie (dodatkowe koszty).
10:00-10:30: Poranny kawiarnia
- Koszt: 5-10 zł
- Opis i wskazówki: Wiele kawiarni w Starym Miastwie oferuje niskie ceny kawy i ciast.

Popołudnie
12:00-14:00: Lunch na plaży
- Koszt: 15-20 zł
- Opis i wskazówki: Wybierz się na lunch na plaży w Gdańsku, aby cieszyć się widokiem na morze. Możesz również poszukać lokalnych barów i restauracji oferujących pyszne dania na przystępnych cenach.

14:00-15:00: Spacer nad brzegiem Morza Bałtyckiego
- Koszt: Bezpłatny
- Opis i wskazówki: Spacer nad brzegiem Morza Bałtyckiego, aby cieszyć się pięknym widokiem na miasto.

Wieczór
18:00-21:00: Spacer po Gdańsku i obiad w lokalnej restauracji
- Koszt: 25-35 zł
- Opis i wskazówki: Spacer po Gdańsku, aby zobaczyć piękne zabytki i miejsca. Możesz również odwiedzić Muzeum Morskie lub Muzeum Gdańska (dodatkowe koszty).

Podsumowanie budżetu
- Łączny koszt: 40-60 zł
- Pozostaje: 40-60 zł

Uwagi:

* Koszty są orientacyjne i mogą się różnić w zależności od lokalizacji i restauracji.
* Transport publiczny w Gdańsku jest dobrze rozwinięty, a bilety kosztują około 2 zł za przejazd.
* W przypadku chęci zwiedzania muzeów, warto sprawdzić ich ceny i godziny otwarcia przed wyjazdem.

==

TEST 3: WARSZAWA - BUDŻET WYSOKI (400 ZŁ)
==
Plan jednodniowego zwiedzania Warszawy (budżet 400 zł)

Rano
8:00-10:00: Muzeum Narodowe
- Koszt: 25 zł (wstęp)
- Opis i wskazówki: Muzeum Narodowe oferuje bogatą kolekcję dzieł sztuki z historii Polski. Można tu zobaczyć obrazy, rzeźby, ceramikę i eksponaty związane z kulturą i historią kraju.
10:00-10:30: Przerwa na kawę i deser
- Koszt: 20 zł (kawa i ciasto)
- Wskazówki: Można wybrać kawiarnię w okolicy Muzeum Narodowego, np. "Cafe Pod Wawelem".

Popołudnie
12:00-15:00: Royal Palace
- Koszt: 10 zł (wstęp)
- Opis i wskazówki: Zobacz królewską historię w Royal Palace. Można zobaczyć rezydencję królów polskich, a także park i ogrody.
15:00-16:00: Spacer po Placu Zamkowym
- Koszt: bezpłatny
- Opis i wskazówki: Plac Zamkowy to serce Warszawy. Spacer po placu pozwala na poznanie historii miasta, a także na odpoczynek.

Wieczór
18:00-21:00: Restauracja "Restauracja Pod Baranem"
- Koszt: 150 zł (porcja)
- Opis i wskazówki: Restauracja Pod Baranem to miejsce, które oferuje tradycyjne polskie dania w przyjaznej atmosferze. Można tu zjeść pierogi, kotlety i inne specjały kuchni pols
kiej.

Podsumowanie budżetu
- Łączny koszt: 250 zł
- Pozostaje: 150 zł

Uwagi:

* Koszty mogą się różnić w zależności od restauracji i usług.
* Plan jest elastyczny, można go modyfikować w zależności od zainteresowań.
* W Warszawie istnieje wiele innych atrakcji, które można odwiedzić, np. Muzeum POLIN, Teatr Wielki, Warszawski Park Zoologiczny.

==

TEST 4: WROCŁAW - BUDŻET ŚREDNI (250 ZŁ) - RODZINA Z DZIEĆMI
==
Wrocław - Plan jednodniowy (budżet 250 zł)

Rano
8:00-10:00: Zwiedzanie Parku Wrocławskiego
* Koszt: Bezpłatny
* Opis i wskazówki: Park Wrocławski to idealne miejsce na rozpoczęcie dnia. Spacer po ścieżkach, obserwowanie zwierząt i oglądanie pięknych widoków na miasto.
* Wskazówka: Przyjdź wczesnym rankiem, aby uniknąć tłumów.

10:00-12:00: Muzeum Wrocławskiego (opcja)
* Koszt: 15 zł (dla dorosłych)
* Opis i wskazówki: Muzeum Wrocławskiego oferuje bogatą kolekcję eksponatów dotyczących historii miasta. Możliwość poznania historii, kultury i sztuki w jednym miejscu.
* Wskazówka: Bilety można kupić online przed przyjazdem, aby uniknąć kolejki.

Popołudnie
12:00-13:00: Lunch na Placu Rynek (opcja)
* Koszt: 25 zł - 40 zł (dla dorosłych)
* Opis i wskazówki: Plac Rynek to serce Wrocławia. Można tu zjeść lunch w lokalnej restauracji, a także podziwiać piękno rynku.

13:00-15:00: Zwiedzanie Katedry św. Mikołaja (opcja)
* Koszt: 10 zł (dla dorosłych)
* Opis i wskazówki: Katedra św. Mikołaja to imponująca budowla, która zachwyci Cię swoją architekturą i bogatą historią.

15:00-16:00: Spacer po centrum Wrocławia
* Koszt: Bezpłatny
* Opis i wskazówki: Zwiedzanie centrum Wrocławia, w tym Placu Rynek, Bazylika Św. Mikołaja i innych zabytków.

Wieczór
18:00-21:00: Spacer po Dolinie Wrocławskiej (opcja)
* Koszt: Bezpłatny
* Opis i wskazówki: Dolina Wrocławskiej to idealne miejsce na spacer po wieczornym mieście, a także na spotkanie z lokalną kulturą.

21:00-22:00: Kolacja w lokalnej restauracji (opcja)
* Koszt: 30 zł - 50 zł (dla dorosłych)
* Opis i wskazówki: Wrocław ma bogatą ofertę restauracji, w których można zjeść smaczną kolację.

Podsumowanie budżetu
- Łączny koszt: 120 zł
- Pozostaje: 130 zł

Uwagi:
* Koszty są orientacyjne i mogą się różnić w zależności od preferencji i wyboru konkretnych atrakcji.
* W przypadku transportu, można skorzystać z komunikacji miejskiej (PKM) lub autobusów.
* Można również skorzystać z aplikacji "Wrocław" lub "Moovit", aby uzyskać szczegółowe informacje o trasach i kosztach przejazdów.

Pamiętaj: Plan jest elastyczny, a każdy może go modyfikować w zależności od swoich preferencji i czasu wolnego.

Wnioski:

Model nie potrafi liczyć, często podawał niepoprawne koszty w stosunku do budżetu jaki jest, źle jest sumował i odejmował kwoty. Nie podał droższego budżetu w żadnym przypadku, aczkolwiek wątpie, że te koszty są poprawne, a zwłaszcza aktualne w dzisiejszych czasach. Model nie ma dostępu do aktualnych danych i nie potrafi ich weryfikować, przez co może podawać nieaktualne lub niepoprawne informacje. Model podążał za schematem zazwyczaj czyli - Zwiedzanie -> Lunch -> Aktywność popołudniowa (zwiedzanie)/Spacer -> Kolacja i tak w większości przypadków. Zawsze było to albo zwiedzanie albo spacer, brak innych atrakcji i aktywności. Model nawet pomylił Kraków z Warszawą podał Wawel i Muzeum Narodowe w przypadku pytania o Warszawę. Podawał ciekawe uwagi jak , transport publiczny jest dobrze rozwinięty i tani, aczkolwiek wszędzie podawał podobną uwagę jak z szablonu. W przypadku niskiego budżetu w Gdańsku podał łączny koszt do 60, pozostały budżet do 60, gdzie całkowity był 80, co jest błędem. Generalnie model radził sobie z generowaniem instrukcji i planów, ale miał trudności z dokładnym liczeniem i weryfikacją informacji finansowych i kreatywnym podejściem do atrakcji i aktywności. Pradowdopobobnie atrakcje kulinarne i konkretne szczegóły co do dań są halucynowane i niepoprawne.

Uwaga: Ograniczenia modeli LLM w operacjach matematycznych wynikają z ich architektury - są to modele językowe, nie kalkulatory. Model nie zawsze pamięta wcześniejsze liczby i nie potrafi ich zawsze dokładnie zsumować lub odjąć. Dlatego czasami się myli w obliczeniach budżetowych. Chociaż pewnie można go zmusić promptem do dokładniejszego liczenia, ale i tak nie będzie to zawsze idealne rozwiązanie.