

# Sprawozdanie Lab 1: Start z lokalnym LLM

## 1. Informacje o środowisku

- **System:** Windows 11
- **CPU:** Intel Core i5-13600K
- **RAM:** 16 GB
- **GPU/VRAM:** NVIDIA GeForce RTX 5080 OC 16 GB
- **Wersja narzędzia:** 0.12.6
- **Nazwy/rozmiary modeli:**
  - qwen2.5:1.5b (1.5B parametrów)
  - SpeakLeash/bielik-1.5b-v3.0-instruct:Q8\_0 (1.5B parametrów)
  - llama3.1:8b (8B parametrów)
  - SpeakLeash/bielik-11b-v2.3-instruct:Q8\_0 (11B parametrów)

## 2. Polecenia uruchamiające

Pierwsze uruchomienie

```
# Instalacja i weryfikacja
ollama --version
ollama list

# Pobranie modeli
ollama pull qwen2.5:1.5b
ollama pull SpeakLeash/bielik-1.5b-v3.0-instruct:Q8_0
ollama pull llama3.1:8b
ollama pull SpeakLeash/bielik-11b-v2.3-instruct:Q8_0

# Pierwsze uruchomienie
ollama run qwen2.5:1.5b
```

Testy funkcjonalne (sekcja 3)

Te testy są wykonane na małych modelach (1.5B parametrów) - To zadanie powinno być wykonane na początku na jak najmniejszych modelach.

**Testy do wykonania:**

1. **Wiedza ogólna:** "Kiedy rozpoczęła się II wojna światowa? Podaj krótki opis wydarzeń."
2. **Instrukcje:** "Napisz instrukcję obsługi nowego smartfona dla początkującego użytkownika."
3. **Kreatywność:** "Napisz wiersz o jesieni."

4. **Tłumaczenie:** "Przetłumacz na angielski: 'Mickiewicz w 'Panu Tadeuszu' pisał: 'Litwo! Ojczyzno moja! ty jesteś jak zdrowie; ile cię trzeba cenić, ten tylko się dowie, kto cię stracił.''''
5. **Ekstrakcja informacji:** "Z tekstu: 'Ala ma kota i psa, a kot kosztował 200 zł, a pies 500 zł' wyodrębnij: {liczba\_zwierząt, rodzaje\_zwierząt, cena\_kota, cena\_psa} i zapisz wynik ekstrakcji w formacie JSON."
6. **Analiza sentymentu:** "Jaki ma sentyment to zdanie? (pozytywny/neutralny/neatywny): 'Świetny pomysł wyjścia bez parasola w deszczowy dzień - teraz jestem kompletnie przemoczony.'"

### Porównanie odpowiedzi modeli:

#### Test 1: Wiedza ogólna

- **qwen2.5:1.5b** - Odpowiedź:

II Wojna Światowa (od II Wielkiej Wojny Zbrojnej) zaczęła się 1 September 1939 roku, gdy Niemcy zajęli Polen i Buzynę na wschód od Paryża.

Opis wydarzeń:

- 1 September 1939: Nazijskie państwa Holandia, Polonia i Bozycka Gwiazda Przeciwścierca (Bulgaria) 签订了《德奥英三国同盟条约》。
- 3 September 1939: Nazijskie państwa 签订了《慕尼黑协议》，以交换波兰领土为条件，放弃了对波兰的侵略企图。
- 1 September 1939: Niemcy zajęli Polen, wycofując się od traktatu z Polską. Hitler odszedł poza granicami polskich.
- 2 September 1939: Jego następcy krewni Ciunijski podjął walkę.
- 6 September 1939: Niemcy zajęli Buzynę, zwojów Paryża i Bruxsę (Bruzgą).

Dlatego sezon rozegrań UEFA Champions League będzie trwał od 24 lipca do 5 sierpnia.

- **SpeakLeash/bielik-1.5b** - Odpowiedź:

II wojna światowa rozpoczęła się 22 czerwca 1941 roku, kiedy Niemcy zaatakowały Związek Radziecki w ramach operacji Barbarossa.

- **Uwagi porównawcze:** Qwen podał poprawną datę rozpoczęcia wojny (1 września 1939), podczas gdy Bielik podał błędную datę ataku na ZSRR (22 czerwca 1941). Odpowiedź Qwen zawierała więcej szczegółów, ale z licznymi błędami faktograficznymi i fragmentami w języku chińskim. Tekst zawiera

nieprawdziwe informacje jak "Naziskie państwa Holandia" oraz kończy się irrelevantną informacją o UEFA Champions League. Bielik podał tylko krótką, ale faktycznie poprawną informację o operacji Barbarossa, jednak pomylił ją z początkiem całej wojny.

## Test 2: Instrukcje

- **qwen2.5:1.5b** - Odpowiedź:

1. Wystarczy, jeśli jesteś sam! Pamiętaj, że nauczenie się jest procesem.
  2. Ucz się korzystać z wyświetlających się informacji: informacje na ekranie lub oknach mogą pomóc Ci zrozumieć funkcje komputera.
  3. Więcej wiedzy o systemie operacyjnym twojego telefonu jest prosto dla Twoich działań i taktyki, co może przyciągnąć do niego więcej korzyści!
  4. Zalecana praca to wykonywanie testów, zanim będziesz zmieniać oryginalny zestaw danych urządzenia.
  5. Próbuj wszystkie możliwości komputera, aby zdobyć wiedzę.
  6. Sprawdzaj regularnie czy twoje telefon jest dostosowany do Twoich potrzeb i preferencji.
  7. Pamiętaj, że nie ma "perfect" nowego telefonu - nie będzie miał wszystkich funkcji, które masz na swoim terenie.
  8. Zawsze pamięta o przewidywaniu twojego ryzyka. Jeśli się stanie coś złego, to jest odpowiedniego miejsca podzielić się z innymi użytkownikami lub sprzedawać telefon.
  9. Bezpieczna obsługa to jedno z największych problemów, co powinien być uwzядony z poważeniem.
  10. Daj sobie czas na naukę! Niektóre rzeczy są proste do zrozumienia, ale może zajmować wiele dni, aby dowiedzieli Ci się wszystkiego.
- Dopiero po kilku testach i przemyślcie powinien być już w stanie korzystać z tego telefonu bez problemów.

- **SpeakLeash/bielik-1.5b** - Odpowiedź:

1. Włącz smartfon, naciskając i przytrzymując przycisk zasilania przez kilka sekund.

- **Uwagi porównawcze:** Model Qwen dostarczył sporo kroków instrukcji, ale one są dość niekonkretne i ogólne, nie skupiając się na konkretnych funkcjach smartfona. Instrukcje są raczej poradami niż praktycznym przewodnikiem. Pomylił telefon z komputerem. Z kolei Bielik podał tylko jeden krok, ale był on jasny i precyzyjny.

### Test 3: Kreatywność

- **qwen2.5:1.5b** - Odpowiedź:

Ciemność śniega, gdzie sięgnął pierścień,  
Jasne światło słońca w szarzy zachodu.  
Rzekę cieplej niż w dniu śniadnym,  
Wiosnę wierci jak oświetlona.

Szafa zbyt długich kieliszków  
Nawet w kąt nie jest wygrywała.  
Nie chce się pożegnać ze smutkiem,  
Jednak wczoraj nie czuł takiego gęstki.

Głos krzyku kobiety na placu,  
W którym śniega nie ma żadnego powodu.  
Paczka zbyt długich kieliszków  
Wsiada do zieleniak, jak w żaweczce.

- **SpeakLeash/bielik-1.5b** - Odpowiedź:

Jesień, jesień, czas na zmiany,  
Liście spadają z drzew, wiatr je niesie.  
Złote i czerwone barwy, w powietrzu zapach,  
Natura przygotowuje się do snu zimowego.

- **Uwagi porównawcze:** Wygląda na to, że oba modele preferują wiersze białe. Qwen stworzył bardziej złożony i abstrakcyjny wiersz, trochę trudny do zrozumienia. Bielik napisał dość prosty wiersz.

### Test 4: Tłumaczenie

- **qwen2.5:1.5b** - Odpowiedź:

Mickiewicz wrote in 'Pan Tadeusz': 'Poland! My homeland! You are like life; how much you should value me, only then will one know who has lost me.'

- **SpeakLeash/bielik-1.5b** - Odpowiedź:

Mickiewicz in 'Pan Tadeusz' wrote: 'Little Poland! My country! You are like health; how much you need to value,

only the one who has lost it will know.

- **Uwagi porównawcze:** Poprawne tłumaczenie wstępu inwokacji:

Lithuania, my country! You are as good health:  
How much one should prize you, he only can tell  
Who has lost you

Oba modele zamienił słowo "Litwa" na "Polska". Wymienne Qwen użył "life" zamiast "health", co zmienia sens oryginału. Bielik podał "health", co jest poprawne. Qwen zmienił strukturę zdania, odniósł się do "me" zamiast "you", co również zmienia sens oryginału. Bielik raczej logicznie zachował strukturę oryginału, chociaż posłużył się bardzo prostym i bezpośrednim językiem. (nie jestem lingwistą, więc nie oceniam dokładnie jakości tłumaczenia pod kątem literackim)

#### Test 5: Ekstrakcja informacji

- **qwen2.5:1.5b** - Odpowiedź:

Poniżej znajduje się skompilowany wynik ekstrakcji w formacie JSON:

```
{  
  "liczba_zwierząt": 2,  
  "rodzaje_zwierząt": ["kot", "pies"],  
  "cena_kota": 200,  
  "cena_psa": 500  
}
```

Znajdziesz podobne dane w formacie JSON, zliczając kwotę za oba zwierzęta.

- **SpeakLeash/bielik-1.5b** - Odpowiedź:

```
{"liczba_zwierząt": 2, "rodzaje_zwierząt": ["kot", "pies"], "cena_kotata": 200,  
 "cena_psa": 500}
```

- **Uwagi porównawcze:** Oba modele poprawnie zidentyfikowały wszystkie istotne informacje, ale Bielik miał problem z poprawnym sformatowaniem JSON-a. Natomiast jest to raczej mało istotny błąd, który nie wpływa na zrozumienie danych.

#### Test 6: Analiza sentymentu

- **qwen2.5:1.5b** - Odpowiedź:

Pozytywny

- **SpeakLeash/bielik-1.5b** - Odpowiedź:

Negatywny.

- **Uwagi porównawcze:** Negatywny jest poprawną odpowiedzią, ponieważ zdanie wyraża frustrację z powodu bycia przemoczonym. Qwen błędnie ocenił sentyment jako pozytywny, co wskazuje na brak zrozumienia kontekstu emocjonalnego zdania. Może to wynikać z tego, że Bielik jest trenowany na języku polskim i lepiej rozumie niuanse emocjonalne w tym języku.

### 3. Tabela porównawcza

GPU

<b>Model</b>	<b>Temperatura</b>	<b>Obserwacje</b>
qwen2.5:1.5b	0.1	Dla 1 pytania generował odpowiedzi po Chińsku. Nie potrafił przetłumaczyć inwokacji (podał po Polsku ponownie), na 2 pytaniu się zawiesił i wyrzucał tokeny w zapętlaając się.
qwen2.5:1.5b	0.5	Podaje więcej informacji ale mniej poprawnych z reguły, lepiej sobie radzi z wymyślaniem obszerniejszych wierszy, podał po Polsku informacje o 2 wojnie ale częściowo niepoprawne. Przestał się zapętać w pytaniu 2.
SpeakLeash/bielik-1.5b-v3.0-instruct:Q8_0	0.1	Tym razem podał poprawną datę rozpoczęcia 2 wojny, poradził sobie z podaniem kilku instrukcji do pytania 2, podał negatywny sentyment, nie potrafił przetłumaczyć zdania na angielski
SpeakLeash/bielik-1.5b-v3.0-instruct:Q8_0	0.5	Podał niepoprawną datę jako 3 wrzesień rozpoczęcia wojny, tym razem zaczął tłumaczyć na język angielski ale niepoprawnie gramatycznie, w podobny sposób podał instrukcję, natomiast źle ją sformatował, dłuższy wiersz
llama3.1:8b	0.1	Podał poprawną datę 2 wojny światowej, ładnie sformatował instrukcję, nie przetłumaczył Litwa -> Lithuania, natomiast tłumaczenie z grubsza ok, poprawnie podał sentyment
llama3.1:8b	0.5	Dość podobnie co do 0.1, podał trochę więcej informacji co do 2 wojny, trochę obszerniejsze instrukcje, o dziwo podał trochę krótszy wiersz (2 strofy zamiast 3 w przypadku 0.1), przetłumaczył Litwa na Lithuania, natomiast tłumaczenie gramatycznie trochę brzmi gorzej, sentyment negatywny

Model	Temperatura	Obserwacje
SpeakLeash/bielik-11b-v2.3-instruct:Q8_0	0.1	W większości wydaje się, że podał poprawne informacje, bardzo szczegółowo rozwinał wydarzenia 2 wojny (choć prośba była o krótkie rozwinięcie), wiersz 4 strofy, sentyment negatywny, bardzo długie instrukcje, tłumaczenie wydaje się gramatycznie ok, przy ekstrakcji podał wyjaśnienie
SpeakLeash/bielik-11b-v2.3-instruct:Q8_0	0.5	Podał tym razem mniej faktów z 2 wojny ale poprawnych (tylko 6 podpunktów w porównaniu do 9 dla temp = 0.1) co do reszty bardzo podobnie

## Krótkie wnioski

- Większe modele radzą sobie znacznie lepiej z zadaniami wymagającymi faktycznej wiedzy i złożonych instrukcji.
- Wyższa temperatura generuje bardziej kreatywne odpowiedzi, ale może prowadzić do błędnych informacji.
- Modele specjalizowane w języku polskim (Bielik) lepiej rozumieją niuanse językowe i emocjonalne w tym języku. Chociaż większy model jak lla3.1:8b radzi sobie całkiem dobrze.
- Modele takie jak Qwen mają tendencję do generowania niepoprawnych informacji i zapętlania się przy niższych temperaturach.
- Wszystkie modele dobrze poradziły sobie z zadaniem ekstrakcji informacji, chociaż formatowanie JSON mogło być lepsze w niektórych przypadkach.
- Szybkość odpowiedzi była błyskawiczna na wszystkich modelach dzięki wydajnemu sprzętowi GPU. Dla porównania małe modele błyskawicznie odpowidały, na CPU troszka wolniej ale akceptowalnie (kilka sekund). Dla dużych modeli były tylko już subtelne różnice w szybkości między GPU, natomiast CPU były strasznie wolne kilkadziesiąt sekund. Oczywiście zależy od ilości tokenów w odpowiedzi, ale tak czy siak znacznie wolniej.
- Modele z większą ilością parametrów (11B) generowały bardziej szczegółowe odpowiedzi, ale czasami były zbyt obszerne w kontekście zadanych pytań.

## 4. (Opcjonalnie) Kod integracji programistycznej

Python + Ollama CLI

```
import subprocess
import json

def ollama_query(model_name, prompt, timeout=60):
    """
    Wysyła zapytanie do modelu Ollama przez CLI

    Args:
        model_name (str): Nazwa modelu (np. "llama3.1:8b")
        prompt (str): Tekst zapytania
        timeout (int): Timeout w sekundach

    Returns:
        dict: Wynik zapytania w formacie JSON
    """
    command = f"ollama query --model {model_name} --prompt '{prompt}' --timeout {timeout}"
    process = subprocess.Popen(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)
    output, error = process.communicate()
    if error:
        raise Exception(f"Error executing command: {error.decode('utf-8')}")
    response = json.loads(output.decode('utf-8'))
    return response
```

```
    str: Odpowiedź modelu lub komunikat błędu
"""

try:
    cmd = ["ollama", "run", model_name]
    res = subprocess.run(
        cmd,
        input=prompt.encode('utf-8'),
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
        timeout=timeout
    )

    if res.returncode == 0:
        return res.stdout.decode('utf-8').strip()
    else:
        return f"Błąd: {res.stderr.decode('utf-8')}"

except subprocess.TimeoutExpired:
    return f"Timeout: Model nie odpowiadał w ciągu {timeout}s"
except Exception as e:
    return f"Błąd wykonania: {str(e)}"

# Przykłady użycia
models = [
    "qwen2.5:1.5b",
    "SpeakLeash/bielik-1.5b-v3.0-instruct:Q8_0",
    "llama3.1:8b",
    "SpeakLeash/bielik-11b-v2.3-instruct:Q8_0"
]

prompt = "Wypisz 3 zalety uczenia przez wzmacnianie, każdą zaletę umieść w jednym zdaniu."

# Test na różnych modelach
for model in models:
    print(f"\n{'='*50}")
    print(f"Model: {model}")
    print(f"{'='*50}")
    response = ollama_query(model, prompt)
    print(response)
    print()
```

Odpowiedzi:

```
=====
Model: qwen2.5:1.5b
=====
1. Zwiększenie efektivności: Uczenie przez wzmacnianie umożliwia automatycznego uczenia się rzeczywistych procesów i prawidłowości, niezaawansej lub często zmieniającej siebie umiejętności.
```

2. Przede wszystkim szybkość: Wystarczy podać do szkoły informacje (teoretyczne instrukcje) i system uczy się z powrotem, bez potrzeby nadawania kolejnych instrukcji lub kontroli.
3. Zrozumienie sytuacji: System uczy się nie tylko na poziomie instrukcyjnym, ale również na poziomie interpretowania i prawidłowego działania w rzeczywistości, co jest szczególnie przydatne w wielu dziedzinach nauki.

=====

Model: SpeakLeash/bielik-1.5b-v3.0-instruct:Q8\_0

=====

A:

1. Uczenie przez wzmacnianie pozwala na szybkie i efektywne uczenie się poprzez nagradzanie pozytywnych zachowań, co prowadzi do szybszego rozwoju umiejętności.
2. Algorytmy uczenia przez wzmacnianie są łatwe w implementacji i mogą być dostosowane do różnych zadań, co czyni je wszechstronnymi narzędziami.
3. Uczenie przez wzmacnianie może być stosowane zarówno w przypadku danych sekwencyjnych (np. sekwencja ruchów), jak i niesekwencyjnych (np. rozpoznawanie obrazów).

=====

Model: llama3.1:8b

=====

Jasne, oto trzy zalety uczenia się poprzez wzmacnianie:

1. Ucząc się poprzez wzmacnianie, dzieci mogą samodzielnie rozwiązać zadania i nabycь doświadczenie.
2. Poprzez wzmacnianie można obserwować postępy uczniów w czasie rzeczywistym.
3. Uczenie się poprzez wzmacnianie pozwala na podejmowanie szybkich korekt, gdy występują niepowodzenia.

=====

Model: SpeakLeash/bielik-11b-v2.3-instruct:Q8\_0

1. **Efektywność w rozwiązywaniu problemów sekwencyjnych**: Uczenie przez wzmacnianie jest szczególnie skuteczne w przypadku problemów, które można przedstawić jako sekwencję decyzji. Dzięki temu podejściu agent uczy się optymalnych działań na podstawie doświadczeń i nagród, co pozwala na efektywne rozwiązywanie złożonych problemów sekwencyjnych.
2. **Adaptacja do zmieniających się warunków**: Uczenie przez wzmacnianie umożliwia agentowi adaptację do dynamicznie zmieniającego się środowiska. Dzięki zdolności do uczenia się na podstawie nowych danych i nagród, agent może dostosowywać swoje działania w czasie rzeczywistym, co jest kluczowe w sytuacjach, gdzie otoczenie nieustannie się zmienia.
3. **Brak konieczności posiadania dużych zbiorów danych**: W przeciwieństwie do tradycyjnych metod uczenia maszynowego, takich jak uczenie nadzorowane, uczenie przez wzmacnianie nie wymaga ogromnych ilości oznaczonych danych treningowych. Agent uczy się poprzez interakcję ze środowiskiem i otrzymywanie nagród za swoje działania, co czyni to podejście bardziej efektywnym w sytuacjach, gdzie dostęp do

dużych zbiorów danych jest ograniczony.

## 5. Pytania o umiejętności modelu

1. **Język polski:** Modele polskie (Bielik) wykazywały lepszą płynność stylistyczną i bardziej naturalny język, ale również popełniały błędy gramatyczne, szczególnie w mniejszych wersjach (1.5B). Qwen2.5:1.5b miał tendencję do przechodzenia na język chiński w trudnych momentach. Większe modele (llama3.1:8b, Bielik 11B) radziły sobie znacznie lepiej z odmianą i fleksją.
2. **Faktyczność:** Najczęstsze halucynacje występowały przy pytaniach historycznych - Bielik 1.5B mylił daty (22 czerwca zamiast 1 września), Qwen generował kompletnie fałszywe informacje o "Nazistowskich państwach Holandia" i kończyło absurdalnie o UEFA Champions League. Większe modele były znacznie bardziej wiarygodne. Ograniczanie halucynacji: niższa temperatura (0.1) czasami pomagała, ale mogła też powodować zapętlanie się modelu (w przypadku generowania instrukcji przez Qwen).
3. **Instrukcje:** Małe modele miały duże trudności - Qwen generował ogólnikowe porady zamiast konkretnych kroków, Bielik 1.5B podawał tylko jeden krok. Większe modele (szczególnie Bielik 11B) radziły sobie doskonale, ale czasami były zbyt obszerne. Numerowane listy i jasna struktura pytań zdecydowanie pomagały w otrzymywaniu lepszych instrukcji.
4. **Rozumowanie:** Myślenie wieloetapowe było widoczne głównie w większych modelach. Bielik 11B przy ekstrakcji informacji nie tylko podał JSON, ale też wyjaśnił proces. Małe modele często nie wykazywały logicznego toku myślenia - Qwen przy analizie sentymentu nie rozpoznał ironii w zdaniu o parasolu. Weryfikowałem to zadaniami wymagającymi kontekstu (ironia, tłumaczenie poezji).
5. **Ekstrakcja informacji:** Wszystkie modele poradziły sobie z podstawową ekstrakcją do JSON, ale były różnice w jakości. Qwen dodawał niepotrzebne komentarze, Bielik 1.5B miał drobny błąd w kluczu ("cena\_kotata" zamiast "cena\_kota"). Większe modele były precyzyjniejsze. Typowe błędy: niepoprawne klucze, dodatkowy tekst poza JSON, problemy z kodowaniem polskich znaków.
6. **Kreatywność vs precyzja:** Temperatura 0.1 dawała bardziej konserwatywne, ale czasem powtarzalne odpowiedzi, jednak przy Qwen powodowała zapętlanie. Temperatura 0.5 zwiększała kreatywność - dłuższe wiersze, więcej szczegółów, ale też więcej błędnych faktów podawanych przez model. Rekomendowane ustawienia temperatury: dla zadań z faktami optymalna wydaje się od 0.0 do 0.3, dla kreatywnych raczej więcej niż 0.5.
7. **Wydajność:** GPU znaczco przyspieszyło generację - modele małe odpowiadały błyskawicznie na GPU vs kilka sekund na CPU. Dla dużych modeli różnica była dramatyczna - kilkadziesiąt sekund na CPU vs sekundy na GPU. Rozmiar modelu miał ogromny wpływ: 11B parametrów generowało znacznie lepsze odpowiedzi niż 1.5B, ale też było zauważalnie wolniejsze.
8. **Bezpieczeństwo:** Nie zaobserwowałem przypadków odmowy odpowiedzi ani cenzury w testowanych scenariuszach - wszystkie pytania były neutralne. Modele nie wykazywały szczególnych mechanizmów bezpieczeństwa przy standardowych pytaniach. Nie testowałem celowo kontrowersyjnych treści.
9. **Porównanie narzędzi:** Dobrze się pracowało z CLI Ollamy, ale raczej wole pracować z API w Pythonie lub innym języku, prawdopodobnie przy serwowaniu modeli lokalnie przez API Ollamy dla większej elastyczności i automatyzacji. Chociaż samo API + Python wystarcza.

## 6. Ostatnia refleksja

Modele lokalne generalnie nie są najlepsze, nie są raczej przydatne do weryfikowania informacji ani generowania faktów, ale mogą być użyteczne do zadań kreatywnych jak pisanie wierszy czy generowanie instrukcji. Parametr temperature + num\_predict mogą być przydatne do różnego typu zadań. Najbardziej wydają się być użyteczne na ten moment do ekstrakcji informacji. Niestety minusem jest wydajność, trzeba mieć dobry sprzęt do bardziej wymagających modeli i trudniejszych zadań, a małe modele są raczej słabe jakościowo. W zasadzie Lokalne LLM na ten moment są raczej dobre do zabawy i eksperymentów.