

Analiza wydajności wybranych modeli językowych na komputerze Raspberry Pi

Karol Paweł Krzykowski*, Piotr Kopciał**
Warszawska Wyższa Szkoła Informatyki, Polska

Streszczenie

W artykule omówiono możliwości implementacji wybranych dużych modeli językowych (LLM) na komputerze Raspberry Pi. W tym celu scharakteryzowano termin „duże modele językowe” (ang. Large Language Models – LLM) oraz omówiono możliwości komputera Raspberry Pi w wersji 5. Przedstawiono koncepcję, realizację oraz wyniki analizy wydajności modeli LLM. Przeanalizowano metryki dotyczące szybkości przetwarzania tokenów i czasu pracy modeli Llama 3.2, Llama 3.1, Phi 3.5 Mini oraz Gemma 2. Zbadano średnie zużycie mocy podczas korzystania z wybranych modeli oraz średnie zużycie mocy potrzebne do wygenerowania jednego tokena przez wybrane modele. Sformułowano wnioski końcowe.

Słowa kluczowe – Raspberry Pi, duże modele językowe, LLM, Llama, Phi 3.5 Mini, Gemma 2, Ollama, Open WebUI

* E-mail: karolkrzykowski@gmail.com

** piotrkopcial@gmail.com

1. Wstęp

Od kilku lat na rynku osiągalnych jest wiele usług różnych firm umożliwiających dostęp do modeli sztucznej inteligencji za pośrednictwem Internetu [1]. Korzystanie z tych usług niesie ze sobą korzyści, ale także zagrożenia i koszty [2]. Poza kosztami takim znaczącym zagrożeniem może być rezygnacja z prywatności. Odpowiedzią na te negatywne zjawiska mogłaby być możliwość uruchomienia dużych modeli językowych na niewielkich, autonomicznych jednostkach obliczeniowych, nawet na własnych urządzeniach. Możliwość uruchomienia wybranych modeli na własnym urządzeniu daje większą kontrolę nad ruchem sieciowym, umożliwiając blokowanie niechcianych połączeń, co pozytywnie przyczynia się do zwiększenia poziomu prywatności podczas konwersacji z wybranym modelem.

Aby sprawdzić, czy istnieje możliwość uruchomienia wybranych modeli LLM na własnym urządzeniu, należało przeprowadzić odpowiedni eksperyment. Taki eksperyment opracowano i przeprowadzono [3]. W jego trakcie zbadano, czy na relatywnie tanim i energooszczędnym komputerze Raspberry Pi można uruchomić wybrane duże modele językowe, zachowując przy tym zadowalającą wydajność, a w dalszej perspektywie testowano możliwość wykorzystania takich urządzeń jako domowego serwera LLM.

Niniejszy artykuł stanowi prezentację wyników uzyskanych w trakcie przeprowadzonego eksperymentu. W kolejnych sekcjach artykułu, w pierwszej kolejności zostanie przedstawiony kontekst badań. W szczególności omówione zostaną modele LLM, a następnie zaprezentowany zostanie komputer Raspberry Pi oraz narzędzia wspomagające korzystanie z modeli LLM. W dalszej części artykułu opisany zostanie przeprowadzony eksperyment badawczy, po czym będą szczegółowo omówione i przedyskutowane uzyskane w eksperymencie wyniki. Na podstawie przeprowadzonej analizy wyników sformułowane zostaną końcowe wnioski.

Osobnego wyjaśnienia wymagają użyte w artykule jednostki pojemności pamięci, takie jak MiB (mebibajty) albo GiB (gibibajty), inne niż powszechnie używane jednostki MB (megabajt) oraz GB (gigabajt). W tych pierwszych jednostkach podstawą jest liczba 2 – pojemności pamięci są opisywane jako potęga liczby 2. W jednostkach drugich podstawą jest liczba 10, a wobec tego pojemności pamięci są przedstawiane jako potęgi liczby 10. Jednostki o podstawie dziesiętnej są przede

wszystkim używane w języku potocznym, w marketingu, w opisie pojemności dysków twardych albo w opisie przepustowości sieci teleinformatycznych. Z kolei, jednostki korzystające z podstawy dwójkowej są głównie wykorzystywane w języku technicznym, np. przy opisie pojemności pamięci RAM. I tak 1 MB (megabajt) to $10^6 = 1000^2$ bajtów. Tymczasem 1MiB (mebibajt) to $2^{20} = 1024^2$ bajtów¹. Analogicznie, 1GB (gigabajt) to $10^9 = 1000^3$ bajtów, zaś 1GiB (gibibajt) to $2^{30} = 1024^3$ bajtów. Podobnie można mówić o jednostkach mniejszych od wspomnianych wyżej, czyli o kB (kilobajtach) i KiB (kibibajtach, a także o jednostkach większych od powyższych, np. tarabajtach (TB) oraz tibibajtach (TiB) [4], [5].

2. Opis kontekstu przeprowadzonych badań

Przeprowadzony eksperyment dotyczył implementacji modeli LLM na komputerze Raspberry Pi. Z jednej strony modele LLM mają swoje niemałe wymagania dotyczące platformy sprzętowej na której będą uruchamiane. Z drugiej strony, komputer Raspberry Pi ma swoje ograniczenia. Implementacja modelu LLM na komputerze Raspberry Pi wymaga pogodzenia ze sobą wspomnianych wymagań implementowanego modelu z ograniczeniami komputera. Z tego powodu istnieje potrzeba przybliżenia zarówno specyfiki modeli LLM jak i specyfiki komputera Raspberry Pi.

a. Modele LLM

Skrót LLM pochodzi od angielskiego *Large Language Models*, co w języku polskim tłumaczy jako „duże modele językowe”. Słowo „duże” w nazwie pochodzi od wielkości zbiorów danych, na których modele te są trenowane [6], [7], [8]. Natomiast sam „model językowy” wskazuje na sposób komunikacji i interakcji. Modele LLM mają również bardzo dużą liczbę parametrów, określaną zazwyczaj w miliardach [9].

Modele LLM pozwalają na generowanie tekstów, takich jak np. treści wiadomości email, tłumaczenie z jednego języka na drugi, streszczenie i przeredagowanie podanego tekstu. Są to jedynie niektóre możliwości LLM. Większość znanych tego

¹ W dalszej części artykułu będą używane obydwa typy jednostek pamięci. Choć w większej liczbie przypadków będą zastosowane jednostki o podstawie dwójkowej (MiB oraz GiB), to w uzasadnionych przypadkach będą także użyte jednostki o podstawie dziesiętnej (GB).

rodzaju modeli obsługuje się w sposób, który przypomina rozmowę przez komunikator tekstowy. Użytkownik zadaje pytanie, czasem w odpowiednim szablonie, po czym model zwraca oczekiwaną odpowiedź.

Ważną rzeczą są także różnice między poszczególnymi modelami LLM. Podczas analizy dostępnych rozwiązań, można zauważyć dwie najbardziej znaczące różnice:

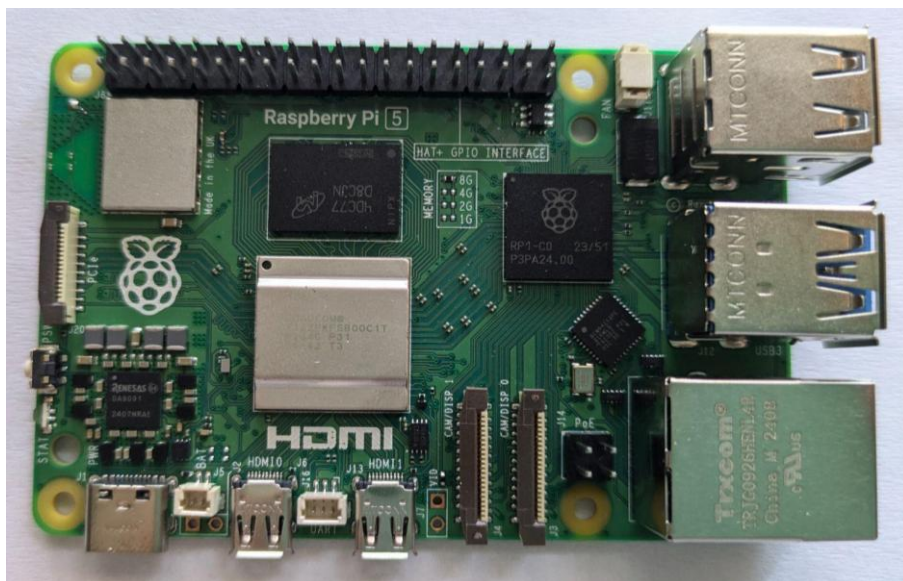
- Pierwszą jest liczba parametrów, często podawana jako liczba z literą ‘b’ na końcu, przykładowo ‘8b’ dla Llama 3:8b. Oznacza to liczbę parametrów wybranego modelu. Wspomniana wcześniej litera oznacza angielski *bilion*, które w naszej, polskiej notacji oznacza *miliard*. Jednymi z najpopularniejszych modeli są te wyposażone w 8 miliardów parametrów,
- Drugą różnicą jest kwantyzacja, polegająca na kompresji modelu poprzez ograniczenie jego precyzji przy jednoczesnym zmniejszeniu jego zapotrzebowania na pamięć. Dzięki temu możliwe jest przyspieszenie trenowania wybranego modelu. Zabieg ten pozwala również na uruchomienie modeli w systemach zawierających mniejszą ilość pamięci niż byłoby to wymagane w przypadku braku kwantyzacji.

Z powodu szeroko wykorzystywanych technik kwantyzacji, mających na celu optymalizację rozmiaru modeli, istotnym elementem różniącym wybrane modele jest liczba parametrów.

b. Raspberry PI

W dalszej części artykułu opisano wykorzystanie mikrokomputera Raspberry Pi 5 (rys. 1). Jego niewielkie rozmiary oraz możliwości rozbudowy o wybrane rozszerzenia, np. takie jak kontrolery silników, sprawiają, że znalazł on swoje zastosowanie w różnorodnych projektach. Komputer ten jest dostępny w kilku wersjach, takich jak Raspberry Pi 1 (Model B), Raspberry Pi 2, 3, 4, aż po najnowszy Raspberry Pi 5. Oprócz nich dostępne są również mniejsze wersje Raspberry Pi Zero lub Compute Module. Głównymi różnicami między nimi są wersje procesorów, ilość pamięci operacyjnej i interfejsy wejścia/wyjścia takie jak porty USB. W przypadku modeli takich jak Raspberry Pi Zero, jest to również sama wielkość mikrokomputera. Dzięki możliwości uruchomienia systemu operacyjnego zgodnego z architekturą ARM, choćby jakiegś dystrybucji Linuxa (np. Raspbian), urządzenie to umożliwia uruchomienie sporej liczby programów działających na komputerach osobistych. Rozszerza to

możliwości tego mikrokomputera umożliwiając jego wykorzystanie zarówno w mniejszych projektach, np. związanych z inteligentnym domem, jak również jako komputer stacjonarny o niskim poborze mocy.



Rysunek 1. Raspberry Pi 5 – komputer wielkości karty kredytowej

Główne różnice wybranego Raspberry Pi 5 względem poprzedniej wersji Raspberry Pi 4, to blisko 3-krotnie szybszy procesor oraz znacznie wydajniejszy układ graficzny [10] [11] [12]. Najnowsza wersja posiada również szybszą pamięć RAM o pojemności do 16 GB. Bardzo istotna w przypadku uruchamiania modeli LLM na tym urządzeniu jest wydajność procesora, gdyż to on zajmuje się obliczeniami. Innym ważnym elementem jest ilość pamięci operacyjnej – w testowanym modelu było to 8 GB. Z perspektywy LLM nie jest to bardzo dużo, gdyż największe modele potrafią wykorzystać kilkakrotność tej wartości.

Komputery Raspberry Pi mają jeszcze jedną istotną zaletę. Ich funkcjonalność można rozszerzać dzięki możliwości stosowania kart rozszerzeń. Jedną z istotnych nowości Raspberry Pi 5 względem poprzedników jest obecność portu PCIe, dzięki któremu można podłączyć dodatkowe urządzenia wspomagające obliczenia dotyczące sztucznej inteligencji.

W kontekście potencjalnych rozszerzeń dla komputera Raspberry Pi, które mogłyby wspomagać jego obliczenia warto wspomnieć o obiecujących doniesieniach dotyczących takich rozszerzeń. W momencie pisania tego artykułu firma Hailo [13] pracowała nad specjalnym modułem obliczeń dla modeli generatywnych [14]. Można mieć nadzieję, że w niedalekiej przyszłości, gdy produkt ten będzie już dostępny, może on znacznie usprawnić działanie LLM na urządzeniu Raspberry Pi.

c. Narzędzia służące do współpracy z LLM

W dalszej części artykułu wykorzystywane są narzędzia służące do współpracy z wybranymi modelami. Pierwszym z nich jest **Ollama**. Jest to oprogramowanie udostępnione w modelu Open Source, służące do uruchamiania wybranych modeli lokalnie, na komputerze użytkownika. Ollama posiada własną bibliotekę modeli, zawierającą również popularne modele takich firm jak Meta lub Google, które można uruchomić za jego pośrednictwem. Interakcja z modelami oferowanymi przez oprogramowanie Ollama odbywa się głównie za pośrednictwem interfejsu terminala. Istnieje również możliwość włączenia serwera, który umożliwi korzystanie z Ollama za pośrednictwem API, z którego to API mogą korzystać inne programy, np. w celu zapewnienia graficznego interfejsu użytkownika [15].

Jednym z wielu programów dostarczających graficzny interfejs użytkownika jest **Open WebUI**. Oprogramowanie to pozwala wybrać dostawcę modeli zarówno kompatybilnych z API firmy OpenAI, jak i ze wspomnianym wcześniej oprogramowaniem Ollama. Sam Open WebUI nie uruchamia modeli, a jedynie zapewnia graficzny interfejs użytkownika i wspiera użytkownika w pisaniu zapytań. W tym celu udostępnia dodatkowe możliwości, takie jak wyszukiwanie w Internecie lub dodawanie wybranych innych źródeł danych, wpływających na dokładniejszą odpowiedź wybranego modelu [16].

3. Implementacja modeli LLM na Raspberry Pi

Analizując wady i zalety rozwiązań zarówno dostępnych w Internecie, jak i uruchamianych lokalnie, można odnieść wrażenie, że największą zaletą korzystania z LLM na własnym urządzeniu jest prywatność. Jest to również powód, dla którego powstaje coraz więcej rozwiązań umożliwiających uruchomienie tych modeli lokalnie.

Domyślną opcją, na jaką stawia większość wymienionych dalej rozwiązań, jest korzystanie z modeli działających na tej samej maszynie, na której będą one wykorzystywane. Jednakże niektóre narzędzia oferują możliwość uruchomienia wybranego modelu w taki sposób, aby był dostępny za pośrednictwem strony internetowej, która np. mogłaby działać na lokalnym serwerze. W przypadku, gdy wykorzystywany przez użytkownika komputer nie posiada wystarczającej ilości zasobów, które może przeznaczyć na uruchomienie wybranego modelu, albo korzysta się z urządzenia, na którym uruchomienie LLM lokalnie nie jest możliwe, to pomysł ten wygląda zachęcająco.

Wobec tego powstaje pytanie, czy można uruchomić taki model na lokalnym serwerze, który będzie działał wystarczająco wydajnie, a jednocześnie będzie zużywał relatywnie mało zasobów? Odpowiedzią może być mały komputer wielkości karty kredytowej, jakim jest Raspberry Pi (rys. 1). Urządzenie to znane jest nie tylko z niewielkich rozmiarów, ale także ze swoich możliwości, które w każdej kolejnej nowej wersji tego urządzenia są coraz większe.

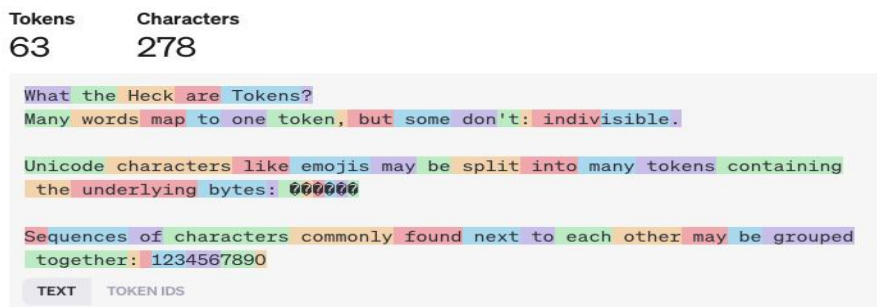
Opisana dalej analiza została przeprowadzona na urządzeniu Raspberry Pi w wersji 5. Powodem takiego wyboru była kwestia wydajności. Była to aktualnie najwydajniejsza i najnowsza wersja tego komputera, umożliwiająca potencjalnie uzyskanie jak najlepszych rezultatów. Najbardziej wymagającymi modelami uruchamianymi na potrzeby opisywanego tu eksperymentu były takie, których liczba parametrów nie przekraczała 8 miliardów. Powyższa decyzja była kompromisem, który z jednej strony uwzględniał oczekiwane możliwości uruchamianego modelu, a z drugiej strony brał pod uwagę ograniczenia na ilość dostępnej pamięci. Dodatkowym powodem była sama wydajność centralnej jednostki obliczeniowej, która przy większych modelach mogła okazać się niewystarczająca.

a. Problemy techniczne

Tutaj należy poruszyć jeszcze dwie kwestie. Pierwszą jest rozwinięcie terminu *token*, który pojawi się w ramach testów wydajności. Jest to najmniejsza jednostka tekstu, którą przetwarza wybrany model. Może to być zarówno pojedyncza litera, jak i część słowa, a nawet całe słowo lub znaki interpunkcyjne (rys. 2). Tekst podawany modelowi musi zostać podzielony na tokeny, gdyż to na ich podstawie model „rozumie” intencje autora, a następnie jest w stanie wygenerować odpowiedzi lub kontynuować

tekst. Wynikiem działania modelu są również tokeny, formujące zrozumiałą dla użytkownika treść. Cały proces związany z tokenizacją tekstu jest potrzebny do prawidłowego funkcjonowania modelu, gdyż przekształca on niezrozumiałą dla sieci neuronowych formę, na postać liczbową. Liczbę generowanych lub przetwarzanych tokenów na sekundę można uznać za miarę szybkości (albo tempa) analizowanych słów lub ich fragmentów [17].

Drugą kwestią jest weryfikacja wykorzystanych zasobów oraz zużycia mocy wykorzystywanej w trakcie realizacji eksperymentu. Do sprawdzania użycia procesora i pamięci operacyjnej, wykorzystano wieloplatformową interaktywną przeglądarkę procesów o nazwie **htop** [18]. Do pomiaru zużywanej mocy wykorzystano watomierz firmy Green Blue [19].



Rysunek 2. Wizualizacja pokazująca podział tekstu na tokeny, wraz z informacją o ich liczbie [20]

4. Opis przeprowadzonego eksperymentu

Przed przejściem do części poświęconej analizie, istotne było ustalenie sposobu jej przeprowadzenia. Zdecydowano, że eksperyment będzie przeprowadzony na dwa sposoby. Pierwszym sposobem była weryfikacja wydajności modelu LLM za pomocą jedynie interfejsu konsolowego Ollama. Drugim sposobem przeprowadzenia eksperymentu była weryfikacja wydajności modelu LLM za pomocą interfejsu graficznego Open WebUI. Należy zauważyć, że w tym drugim przypadku interfejs Open WebUI ograniczy także dostępne zasoby obliczeniowe.

W celu sprawdzenia wydajności wybranych modeli, zbierano informacje dotyczące szybkości przetwarzania zapytania użytkownika, czasu generowania odpowiedzi, a także czasu trwania całości procesu. Dodatkowo badano zużycie mocy podczas pracy. Aby uzyskać najlepsze rezultaty, wybrane modele były pytane o to samo 5 razy, a następnie obliczano wartość średnią z wyników.

Wszystkie modele były uruchamiane bezpośrednio z interfejsu konsolowego Ollama, z wykorzystaniem odpowiedniego argumentu prezentującego na koniec procesu dane wykorzystane w analizie. Również jeden z modeli uruchamiano wykorzystując interfejs graficzny Open WebUI.

Do przeprowadzenia analizy wybrano jedno, takie samo polecenie, które było wykonane na każdym z wymienionych wcześniej modeli. Z powodu wykorzystywania różnych rozwiązań, niekiedy posługujących się ograniczoną liczbą języków, zapytanie to sformułowano w języku angielskim. Wybrany pytaniem, jakie było zadawane każdemu modelowi, była prośba podania przepisu na babeczki. Decyzję tę można argumentować jako jedną z wielu potencjalnych możliwości wykorzystania modeli LLM. Zapytanie w języku angielskim posiada następującą treść: *Write me the recipe for cupcakes* [21].

a. Analiza wydajności modeli LLM współpracujących ze środowiskiem Ollama

Po objaśnieniu sposobu przeprowadzenia analizy można przejść dalej, czyli do weryfikacji wydajności modeli działających w oparciu o oprogramowanie Ollama. Dzięki dostarczonym przez ten program danym, takim jak szybkość przetwarzania i długość wykonywania operacji, można było zweryfikować wydajność wybranych modeli i oszacować czy można ją uznać za wystarczającą.

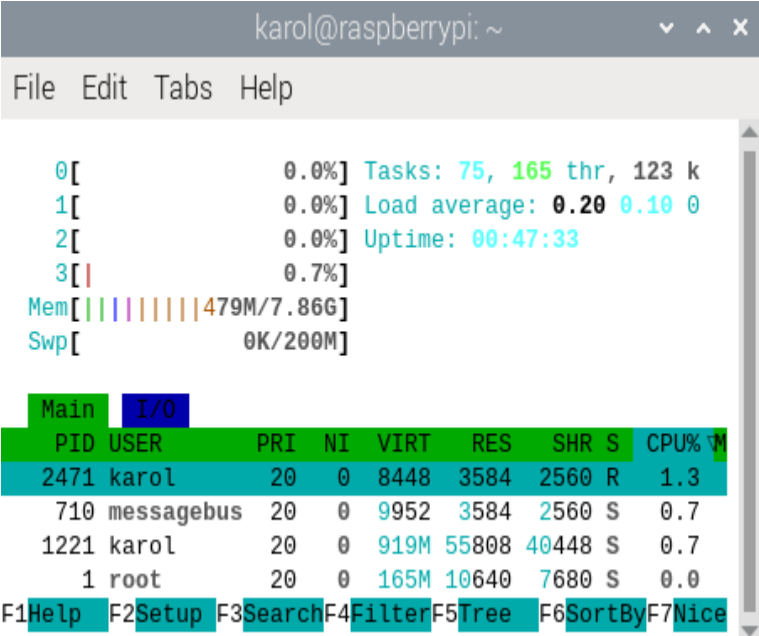
Przed rozpoczęciem testów, przy działającym jedynie programie „htop”, wykonano pomiary zużycia mocy i zasobów systemu. Wykorzystanie procesora zmieniało się w zakresie od 0,7% do 5%. Natomiast poziom zużycia pamięci RAM oscylował w pobliżu 479 MiB przy możliwym dostępnym poziomie 7,86 GiB (rys. 3).

W celu sprawdzenia poboru mocy wykorzystano watomierz firmy Green Blue. Wartości przez niego wskazywane były weryfikowane podczas pracy modeli. Pomiary dla urządzenia bez żadnego obciążenia wyniósł jedynie 3,8 watów mocy.

Rozpoczynając badania założono, że zostaną zbadane następujące modele:

- Llama 3.2 z miliardem parametrów,
- Llama 3.1 z ośmioma miliardami parametrów,
- Phi 3.5 z trzema miliardami ośmiuset milionami parametrów,
- Gemma 2 z dwoma miliardami parametrów.

Po zakończeniu badania wybranego modelu, dla każdego z testowanych parametrów jako wynik testu brano jego wartość średnią uzyskaną w pięciu pomiarach.



The screenshot shows a terminal window titled 'karol@raspberrypi: ~'. It displays system statistics and a process list. The statistics include tasks, load average, uptime, memory usage (479M/7.86G), and swap usage (0K/200M). The process list is a table with columns: Main, I/O, PID, USER, PRI, NI, VIRT, RES, SHR, S, CPU%, and %MEM. The processes listed are karol (PID 2471), messagebus (PID 710), karol (PID 1221), and root (PID 1). At the bottom, there are function key shortcuts: F1Help, F2Setup, F3Search, F4Filter, F5Tree, F6SortBy, and F7Nice.

```
karol@raspberrypi: ~
File Edit Tabs Help

0[          0.0%] Tasks: 75, 165 thr, 123 k
1[          0.0%] Load average: 0.20 0.10 0
2[          0.0%] Uptime: 00:47:33
3[|         0.7%]
Mem[|||||479M/7.86G]
Swp[         0K/200M]

Main I/O
PID USER    PRI  NI  VIRT   RES   SHR  S  CPU% %MEM
2471 karol    20   0  8448   3584   2560 R   1.3  0.7
 710 messagebus 20   0  9952   3584   2560 S   0.7  0.7
1221 karol    20   0  919M  55808  40448 S   0.7  0.7
   1 root      20   0  165M  10640   7680 S   0.0  0.0
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice
```

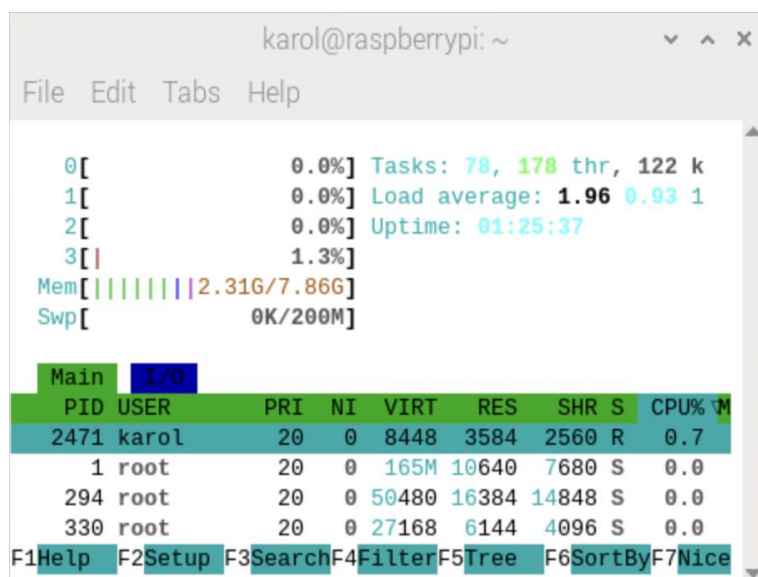
Rysunek 3. Konsola systemu Raspberry Pi – oprogramowanie htop

Przed każdym kolejnym testem, oprogramowanie Ollama było uruchamiane ponownie w celu wyczyszczenia pamięci podręcznej, mogącej przechowywać przetworzone wcześniej dane, mogąc w ten sposób wpłynąć na wyniki analizy.

Llama 3.2

Jako pierwszy model LLM badano model Llama 3.2 firmy Meta, posiadający 1 miliard parametrów. Analizę rozpoczęto od wykonania podanego w terminalu polecenie: „Ollama run llama 3.2:1b – verbose”. W poleceniu tym istotny jest podany na końcu

argument „verbose”. Dzięki niemu jest ono wykonywane w trybie pracy, który wyświetla szczegółowe informacje dotyczące szybkości przetwarzania tokenów oraz czasu wykonania tego polecenia.



Rysunek 4. Konsola systemu Raspberry Pi – oprogramowanie htop przy załadowanym modelu Llama 3.2

Po pojawieniu się konsolowego interfejsu komunikacji z modelem, podane zostało polecenie testowe. Tuż po załadowaniu modelu, ale jeszcze przed rozpoczęciem pracy, wykorzystanie pamięci operacyjnej wzrosło do 2,3 GiB, natomiast procent wykorzystania procesora nie zmienił się (rys. 4). Podczas generowania odpowiedzi, wykorzystanie procesora osiągnęło poziom 100%, a wykorzystanie pamięci operacyjnej pozostało na poziomie prawie nie zmienionym (2,31GiB). W tym samym czasie pobór mocy wzrósł do poziomu 11 watów.

Analizowany model czasami nie był w stanie odpowiedzieć na postawione mu pytanie o przepis na babeczki, gdyż uznawał je za niezgodne z zasadami bezpieczeństwa. W pozostałych przypadkach model udzielał odpowiedzi zgodnie z oczekiwaniami (rys. 5). Wartości uzyskiwane w testach zostały przedstawione w tabeli 1.

```
karol@raspberrypi: ~
File Edit Tabs Help

liners.
6. Bake for 18-20 minutes, or until a toothpick
inserted into the center of a cupcake comes out
clean.
7. Allow the cupcakes to cool completely in the
pan.

Optional:

* For a chocolate flavor, add 1-2 tablespoons of
unsweetened cocoa powder to the dry ingredients.
* To make red velvet cupcakes, use buttermilk
instead of regular milk and add 1 teaspoon of
red food coloring to the batter.
* To make gluten-free cupcakes, substitute the
all-purpose flour with a gluten-free flour
blend.

Tips:

* Make sure to use room temperature ingredients
for the best results.
* Don't overmix the batter, as this can lead to
tough cupcakes.
* If you're using a different type of milk, such
as almond or soy milk, you may need to adjust
the liquid content of the recipe accordingly.

total duration:      50.743398368s
load duration:       42.220341ms
prompt eval count:   31 token(s)
prompt eval duration: 1.656875s
prompt eval rate:    18.71 tokens/s
eval count:          367 token(s)
eval duration:       48.998589s
eval rate:           7.49 tokens/s
>>> Send a message (? for help)
```

Rysunek 5. Konsola systemu Raspberry Pi – jedna z odpowiedzi udzielonych przez model Llama 3.2, wraz z podanymi metrykami

Tabela 1. Metryki dotyczące szybkości przetwarzania tokenów i czasu pracy modelu Llama 3.2

Numer testu	Szybkość przetwarzania tokenów zapytania na sekundę	Szybkość przetwarzania tokenów odpowiedzi na sekundę	Łączny czas pracy wyrażony w sekundach	Łączna liczba tokenów pytania i odpowiedzi
1	18,71	7,49	50,74	398
2	18,70	7,38	67,35	515
3	18,64	7,36	64,91	496
4	19,00	7,34	60,70	464
5	19,18	7,45	54,15	422
Wartość średnia	18,85	7,40	59,57	459

Llama 3.1

Kolejnym analizowanym modelem był model Llama 3.1, posiadający o miliardy parametrów, również opracowany także przez firmę Meta. Zauważono, że model ten wykorzystywał więcej pamięci operacyjnej niż badany wcześniej model Llama 3.2. Ten fakt wynika z jego większej złożoności: model Llama 3.1 posiada prawie ośmiokrotnie więcej parametrów niż model Llama 3.2. Zużycie pamięci RAM podczas korzystania z modelu wynosiło około 5,68 GiB, dodatkowo system zaczął wykorzystywać dostępną przestrzeń wymiany (ang. swap space) (rys. 6).

Zużycie mocy po załadowaniu modelu było takie samo jak przy braku obciążenia. Podczas generowania odpowiedzi również nieznacznie wzrosło. Raspberry Pi pobierał wtedy moc o wartości około 12,7 watów. Wydajność modelu była odpowiednio niższa niż w przypadku Llama 3.2. Subiektywne odczucie również potwierdza znacznie wolniejszy czas odpowiedzi. Ten czas odpowiedzi sprawia, że korzystanie z modelu staje się niekomfortowe. Wartości uzyskane w testach tego modelu również zostały zaprezentowane w tabeli 2.

```
karol@raspberrypi: ~
File Edit Tabs Help

0[|||||||100.0%] Tasks: 74, 166 thr, 126 k
1[|||||||100.0%] Load average: 2.98 1.39 1
2[|||||||99.3%] Uptime: 00:18:42
3[|||||||100.0%]
Mem[|||||||5.68G/7.86G]
Swp[|||||||192M/200M]

Main I/O
PID USER PRI NI VIRT RES SHR S CPU%
2988 ollama 20 0 6679M 5537M 5120 R 397.4 6
3190 ollama 20 0 6679M 5537M 5120 R 33.7 6
3191 ollama 20 0 6679M 5537M 5120 R 33.0 6
3189 ollama 20 0 6679M 5537M 5120 R 32.3 6
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice
```

Rysunek 6. Konsola systemu Raspberry Pi – oprogramowanie htop podczas pracy modelu Llama 3.1

W wyniku przeprowadzonych testów, Llama 3.1 z 8 miliardami parametrów wykazuje możliwość przetwarzania średnio 2,46 tokenów na sekundę w przypadku pytania i 1,91 tokenów w przypadku odpowiedzi. Średni łączny czas pracy wyniósł 317,27 sekund. Biorąc pod uwagę liczbę analizowanych przez model tokenów, wynoszącą średnio 582,8, można obliczyć, że model Llama 3.1 był w stanie przetworzyć średnio 1,84 tokenów na sekundę. Wydajność tego modelu na Raspberry Pi jest za niska. Zarówno subiektywne odczucia, jak i dane mówiące o szybkości generowania treści wskazują na niezadowalającą szybkość działania. Model potrzebuje zbyt dużo czasu, aby wygenerować odpowiedź w czasie zadowalającym użytkownika.

Tabela 2. Metryki dotyczące szybkości przetwarzania tokenów i czasu pracy modelu Llama 3.1

Numer testu	Szybkość przetwarzania tokenów zapytania na sekundę	Szybkość przetwarzania tokenów odpowiedzi na sekundę	Łączny czas pracy wyrażany w sekundach	Łączna liczba tokenów pytania i odpowiedzi
1	2,47	1,90	395,91	631
2	2,45	1,89	311,10	592
3	2,46	1,92	250,84	484
4	2,46	1,92	326,62	631
5	2,46	1,90	301,89	576
Wartość średnia	2,46	1,91	317,27	582,8

Phi 3.5 Mini

Następnym badanym modelem był model Phi 3.5 Mini firmy Microsoft. Model ten posiada 3,8 miliardów parametrów, dlatego potrzebował mniej pamięci operacyjnej. W tym przypadku zużycie pamięci RAM podczas korzystania z modelu wynosiło około 5,34 GiB. Zużycie to jest niewiele niższe od zużycia pamięci w przypadku większego modelu Llama 3.1 (rys. 7). Zużycie mocy natomiast było o około 1 wat niższe niż w przypadku modelu Llama 3.1 podczas generowania odpowiedzi i wynosiło około 11,6 watów. Przy samym załadowanym modelu, wartość ta się nie zmieniła względem stanu bezczynności. Model Phi 3.5 Mini wykazywał się podczas pracy wyższą szybkością działania niż Llama 3.1. Jednakże w subiektywnym odczuciu nadal była to niewystarczająca szybkość, aby uznać działanie tego modelu za wystarczająco wydajne. Potwierdzają to dane zamieszczone w tabeli 3.

```
karol@raspberrypi: ~  
File Edit Tabs Help  
  
0[|||||||98.7%] Tasks: 74, 164 thr, 129 k  
1[|||||||100.0%] Load average: 1.46 2.94 3  
2[|||||||99.3%] Uptime: 01:23:57  
3[|||||||99.3%]  
Mem[|||||||5.33G/7.86G]  
Swp[|||||||181M/200M]  
  
Main 170  
PID USER PRI NI VIRT RES SHR S CPU% M  
13848 ollama 20 0 6314M 5165M 5120 R 390.8 6  
13917 ollama 20 0 6314M 5165M 5120 R 6.6 6  
13918 ollama 20 0 6314M 5165M 5120 R 6.6 6  
13919 ollama 20 0 6314M 5165M 5120 R 5.9 6  
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice
```

Rysunek 7. Konsola systemu Raspberry Pi – oprogramowanie htop podczas pracy modelu Phi 3.5 Mini

Tabela 3. Metryki dotyczące szybkości przetwarzania tokenów i czasu pracy modelu Phi 3.5 Mini

Numer testu	Szybkość przetwarzania tokenów zapytania na sekundę	Szybkość przetwarzania tokenów odpowiedzi na sekundę	Łączny czas pracy wyrażany w sekundach	Łączna liczba tokenów pytania i odpowiedzi
1	4,64	2,33	786,66	1841
2	4,67	2,76	403,33	1120
3	4,66	2,70	445,93	1212
4	4,64	2,50	607,56	1527
5	4,68	2,76	402,63	1087
Wartość średnia	4,66	2,61	529,22	1357,4

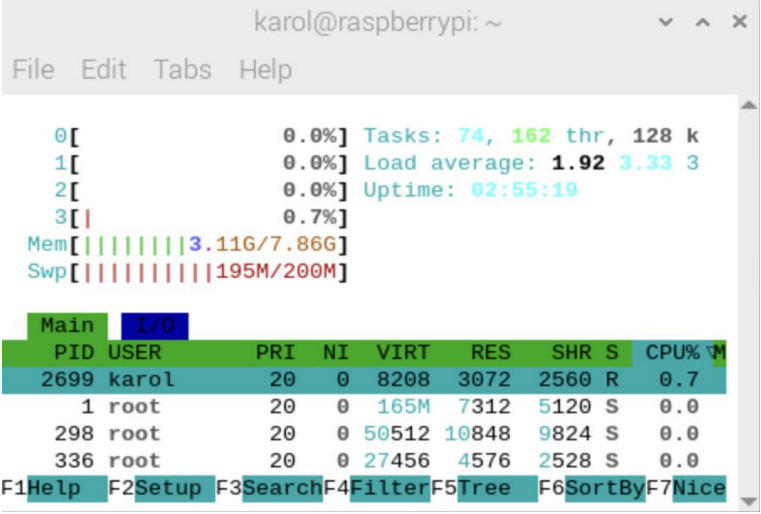
W wyniku przeprowadzonych testów, Phi 3.5 Mini z 3.8 miliardami parametrów wykazuje możliwość przetwarzania średnio 4,66 tokenów na sekundę w przypadku

pytania i 2,61 tokenów w przypadku odpowiedzi. Średni łączny czas pracy wyniósł 529,22 sekund. Biorąc pod uwagę liczbę analizowanych przez model tokenów, wynoszącą średnio 1357,4, można obliczyć, że model Phi 3.5 Mini był w stanie przetworzyć średnio 2,56 tokenów na sekundę.

Wydajność tego modelu na Raspberry Pi jest wprawdzie lepsza niż wydajność wcześniej analizowanego Llama 3.1, jednak wciąż pozostaje niezadowalająca. Phi 3.5 Mini zarówno w subiektywnym odczuciu, jak i według przedstawionych danych, nie nadaje się do prowadzenia konwersacji przy pomocy urządzenia Raspberry Pi. W przypadku, gdyby model był w stanie generować odpowiedź z szybkością 3,5 tokenów na sekundę, wówczas można by określić jego wydajność jako wystarczającą podczas dyskusji z użytkownikiem.

Gemma 2

Ostatnim badanym modelem był model Gemma 2 firmy Google, posiadający 2 miliardy parametrów. Był to drugi co do wielkości analizowany model. Zużycie pamięci RAM podczas korzystania z tego modelu wynosiło około 3,11 GiB i było nieco wyższe niż zużycie pamięci przez model Llama 3.2 (rys. 8). Pobór mocy był z kolei wyższy niż w przypadku analizowanego wcześniej modelu Phi 3.5 Mini. Odczytywane na watomierzu wartości mierzonej mocy oscylowały w okolicach 12 watów.



```
karol@raspberrypi: ~
File Edit Tabs Help

0[          0.0%] Tasks: 74, 162 thr, 128 k
1[          0.0%] Load average: 1.92 3.33 3
2[          0.0%] Uptime: 02:55:19
3[|         0.7%]
Mem[|||||||3.11G/7.86G]
Swp[|||||||195M/200M]

Main  I/O
PID USER    PRI  NI  VIRT  RES  SHR S  CPU% M
2699 karol   20   0  8208  3072 2560 R   0.7
   1 root     20   0  165M  7312 5120 S   0.0
  298 root     20   0  50512 10848 9824 S   0.0
  336 root     20   0  27456  4576 2528 S   0.0
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice
```

Rysunek 8. Konsola systemu Raspberry Pi – oprogramowanie htop przy załadowanym modelu Gemma 2

Model Gemma 2 firmy Google dostarczał subiektywnie zadowalającą szybkość generowania tokenów. Powyższa obserwację potwierdzają również dane przedstawione w tabeli 4.

Tabela 4. Metryki dotyczące szybkości przetwarzania tokenów i czasu pracy modelu Gemma 2

Numer testu	Szybkość przetwarzania tokenów za pytania na sekundę	Szybkość przetwarzania tokenów od odpowiedzi na sekundę	Łączny czas pracy wyrażony w sekundach	Łączna liczba tokenów pytania i odpowiedzi
1	8,26	4,52	129,06	575
2	8,18	4,49	129,46	588
3	8,23	4,42	159,91	713
4	8,16	4,44	150,68	676
5	8,11	4,50	131,18	597
Wartość średnia	8,19	4,47	140,06	629,8

W wyniku przeprowadzonych testów, Gemma 2 z 2 miliardami parametrów wykazuje możliwość przetwarzania średnio 8,19 tokenów na sekundę w przypadku pytania i 4,47 tokenów w przypadku odpowiedzi. Średni łączny czas pracy wyniósł 140,06 sekund. Biorąc pod uwagę średnią liczbę tokenów analizowanych przez model (629,8) oraz średni czas pracy (140,06 sekund), można obliczyć, że model Gemma 2 mógł przetworzyć średnio 4,5 tokenów na sekundę.

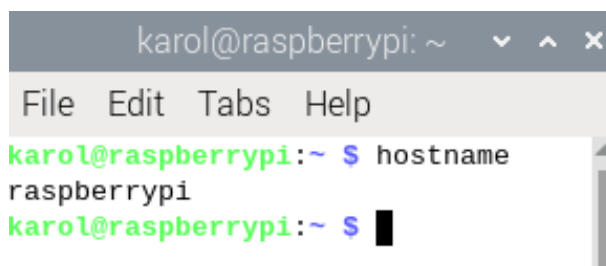
Wydajność tego modelu na Raspberry Pi jest zadowalająca. Szybkość generowania tekstu powoduje, że czas oczekiwania podczas generowania kolejnych tokenów odpowiedzi jest subiektywnie satysfakcjonujący. Model ten można z powodzeniem wykorzystać do zadań związanych z konwersacją z użytkownikiem. Również jego wydajność jest niewiele niższa od wydajności, jaką można uznać za wystarczającą.

b. Analiza wydajności modelu LLM współpracującego z graficznym interfejsem Open WebUI

Po przeprowadzonej analizie wydajności wybranych modeli z użyciem oprogramowania Ollama, można było przejść do kolejnego etapu, polegającego na sprawdzeniu szybkości działania podczas korzystania z wybranego wcześniej interfejsu graficznego. Powinien on pozwolić na korzystanie z wybranego modelu w prosty i przejrzysty dla użytkownika sposób.

Ponieważ interfejs graficzny Open WebUI współpracuje z oprogramowaniem Ollama, dlatego istotne było, aby upewnić się, czy zarówno oprogramowanie Ollama, jak i Open WebUI są prawidłowo skonfigurowane. W przeciwnym wypadku narzędzia te mogą siebie nawzajem „nie widzieć”, co może doprowadzić do braku możliwości korzystania z interfejsu w prawidłowy sposób.

Aby uzyskać dostęp do interfejsu Open WebUI z poziomu wybranego przez siebie urządzenia, należało najpierw albo sprawdzić adres urządzenia w sieci, albo jego nazwę. To drugie można było wykonać, wpisując polecenie „hostname” w terminalu tego urządzenia (rys. 9).

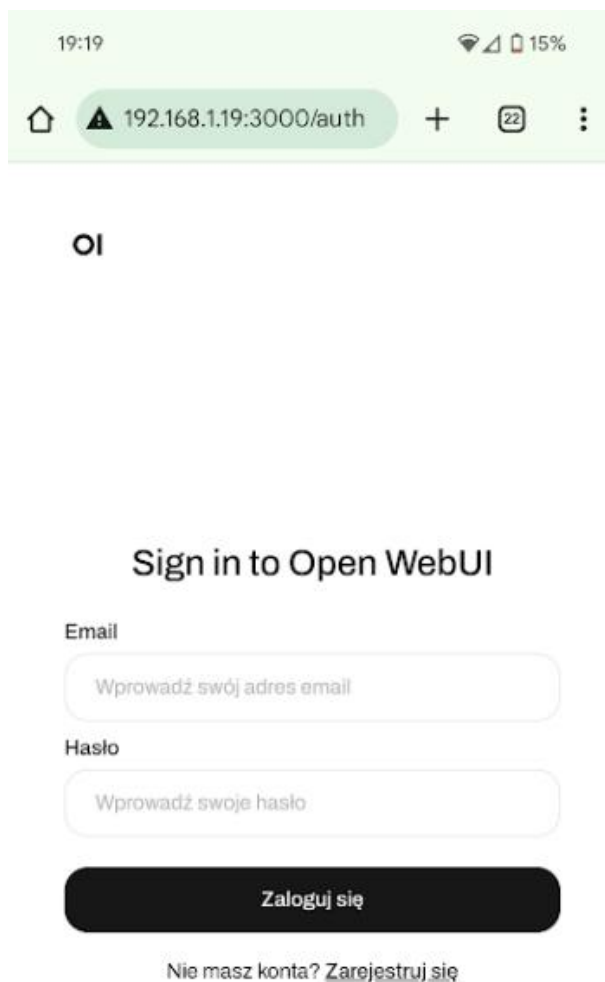


```
karol@raspberrypi: ~  
File Edit Tabs Help  
karol@raspberrypi:~ $ hostname  
raspberrypi  
karol@raspberrypi:~ $
```

Rysunek 9. Konsola systemu Raspberry Pi – wynik wykonania polecenia hostname

W przypadku domyślnej instalacji systemu Raspberry Pi, urządzenie to powinno mieć nazwę „raspberrypi”. Aby uzyskać dostęp do interfejsu, należało w przeglądarce wpisać adres `http://raspberrypi:3000`. Zamiast nazwy urządzenia, można było wykorzystać również jego adres w sieci lokalnej (rys. 10). Teraz powinien pojawić się ekran logowania (rys. 10). W związku z tym, że do skorzystania z interfejsu potrzebne jest ponowne pobieranie modeli, dlatego przed rozpoczęciem dalszej analizy należało wejść do ustawień aplikacji w panelu administracyjnym i znaleźć zakładkę

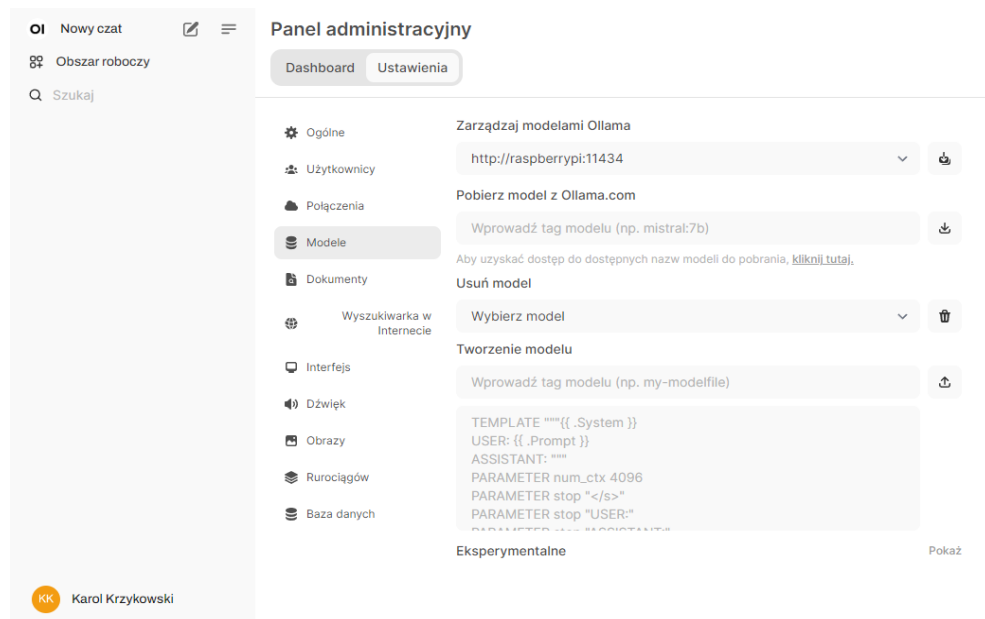
o nazwie „Modele”, a następnie należało pobrać model, z którego będzie się korzystać, poprzez podanie odpowiedniego tagu tego modelu ze strony internetowej [22] (rys. 11). W niniejszym eksperymencie badano tylko model Llama 3.2 w wersji z miliardem parametrów



Rysunek 10. Fragment ekranu logowania Open WebUI na urządzeniu mobilnym, do którego wyświetlenia wykorzystano adres urządzenia w sieci lokalnej

Aby stwierdzić, czy wykorzystanie interfejsu graficznego Open WebUI znacząco zmieni wydajność badanego modelu, po zakończeniu badania, wyniki uzyskane w badaniu modelu z użyciem interfejsu graficznego Open WebUI były porównane z

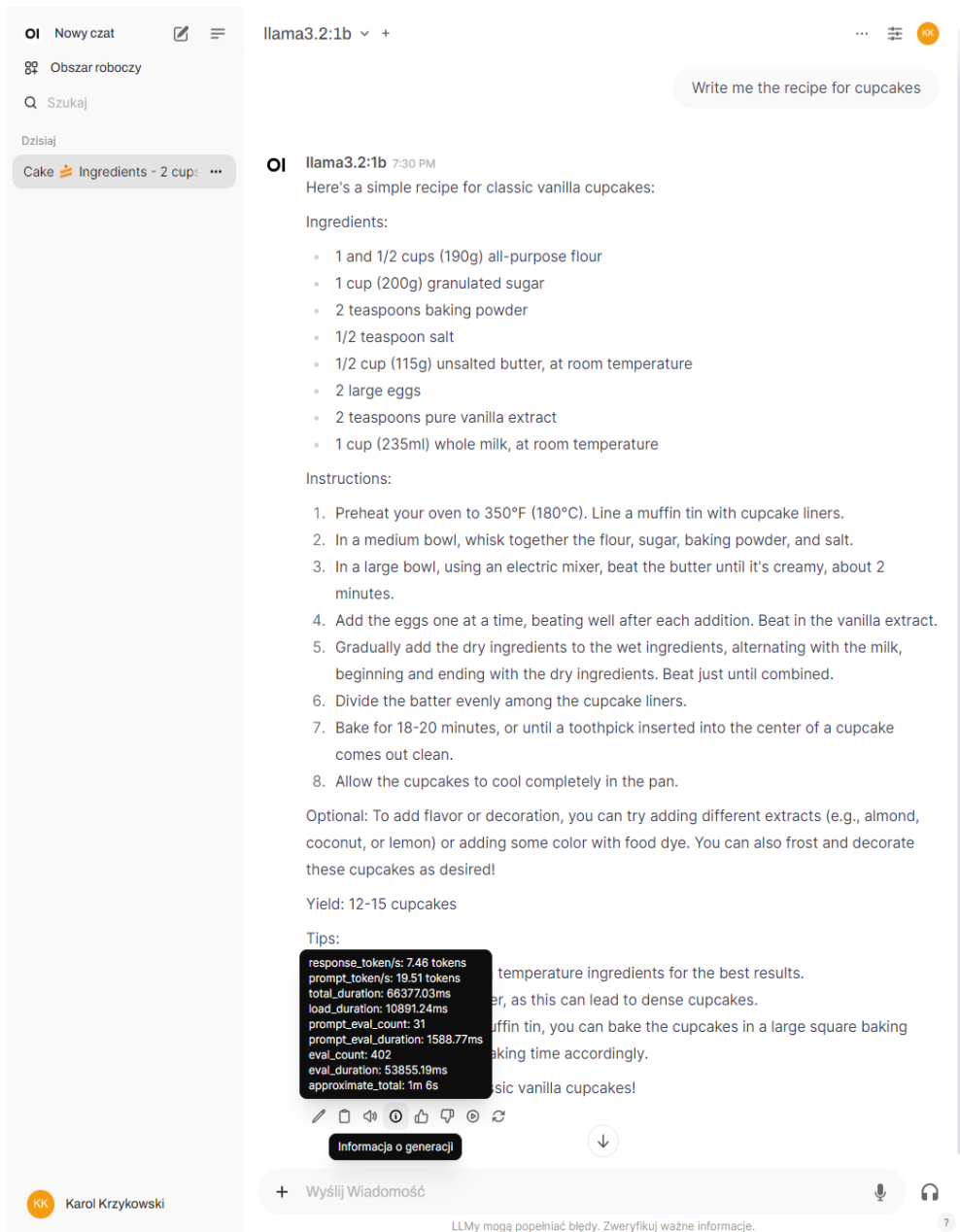
wynikami dla tego samego modelu uzyskanymi w badaniu przeprowadzonym bez interfejsu graficznego Open WebUI, a wyłącznie przy pomocy oprogramowania Ollama.



Rysunek 11. Panel administracyjny Open WebUI, zakładka Modele

Przed rozpoczęciem badania, istotne było również zmierzenie poboru mocy przez urządzenie. Dla uruchomionego serwera Ollama oraz aplikacji Open WebUI, wartość ta wynosi 3,9 watów. Jest to nieznacznie więcej niż w stanie bezczynności.

Podobnie jak w poprzednich przypadkach, tutaj również przeprowadzono 5 testów, polegających na zadaniu pytania dotyczącego przepisu na babeczki. Każde kolejne zapytanie było wykonane w oddzielnej konwersacji, w celu uniknięcia wykorzystywania zapamiętanych przez model elementów wspólnych. Wykorzystanymi metrykami dotyczącymi generowania były metryki znajdujące się pod wygenerowaną odpowiedzią (rys. 12). Dane dotyczące wydajności modelu wykorzystywanego przez interfejs Open WebUI zostały umieszone w tabeli 5.



Rysunek 12. Panel konwersacji z modelem w aplikacji Open WebUI

Tabela 5. Metryki dotyczące szybkości przetwarzania tokenów i czasu pracy modelu Llama 3.2 przy wykorzystaniu Open WebUI

Numer testu	Szybkość przetwarzania tokenów zapytania na sekundę	Szybkość przetwarzania tokenów odpowiedzi na sekundę	Łączny czas pracy wyrażany w sekundach	Łączna liczba tokenów pytania i odpowiedzi
1	19,51	7,46	66,38	433
2	19,64	7,36	61,22	454
3	51,14	7,19	66,61	505
4	52,40	7,16	64,24	486
5	52,59	7,07	70,65	526
Wartość średnia	39,06	7,25	65,82	480,8

Na podstawie danych z tabeli 5 można zauważyć, że już w trzecim teście nie udało się uniknąć pewnego rodzaju optymalizacji, polegającej prawdopodobnie na wykorzystaniu pamięci podręcznej przez Open WebUI w celu zapisania powtarzalnych części zapytania, dzięki czemu szybkość przetwarzania zapytania znacząco wzrosła. Nie wpłynęło to natomiast na czas generowania odpowiedzi. W tym wypadku można zaobserwować pewien trend, w ramach którego szybkość przetwarzania tokenów stopniowo spadała. Może to być podyktowane wysokimi temperaturami urządzenia, mimo umiejscowienia go w dobrze wentylowanej przestrzeni.

W związku z istotnie różnymi szybkościami przetwarzania zapytania, podanie średniej wartości uzyskiwanej dla szybkości przetwarzania zapytania nie jest wiarygodne. Z drugiej strony, wyniki pomiarów uzyskane dla pozostałych metryk dotyczących szybkości przetwarzania tokenów i czasu pracy modelu mogą zostać przedstawione za pomocą ich średnich wartości. Dla szybkości generowania odpowiedzi, wartość ta wynosi 7,25 tokenów na sekundę. Średni czas pracy wyniósł 65,82 sekundy. Średnia liczba przetwarzanych tokenów wyniosła 480,8. Na podstawie

dwóch ostatnich wartości można obliczyć średnią liczbę generowanych tokenów na sekundę. Wynosi ona 7,3. Jest to o 0,41 mniej niż w przypadku testów w programie Ollama, gdzie wynik ten wynosił 7,71. W przeliczeniu na procenty, model Llama 3.2 wykorzystywany przez interfejs Open WebUI oferował 94,68% wydajności, w stosunku do przypadku, gdy był uruchamiany jedynie poprzez aplikację Ollama.

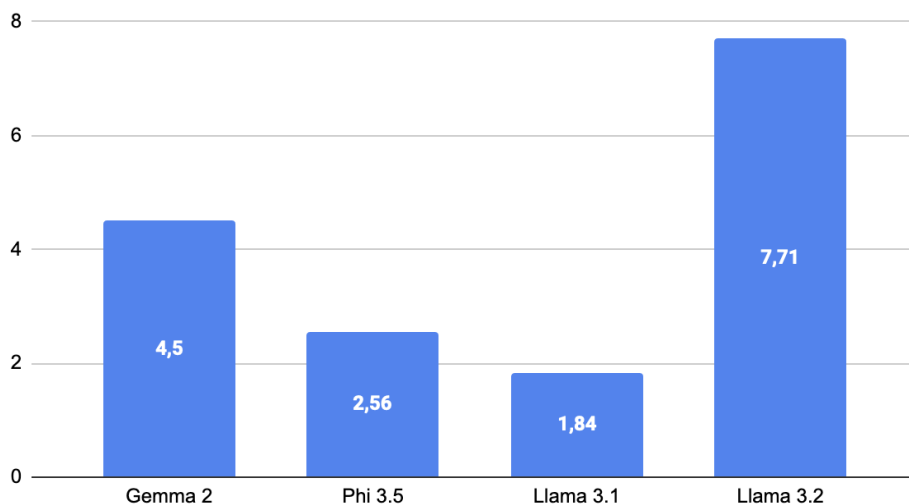
Z kolei, pobór mocy podczas pracy wynosił około 10,5 watów. Wartość ta była mniejsza o 0,5 wata niż w przypadku pracy z modelem działającym tylko z oprogramowaniem Ollama. Powodem takiej sytuacji może być działający na tym samym urządzeniu Open WebUI. Program ten może wykorzystywać dodatkowo procesor w celu wysyłania fragmentów odpowiedzi do klienta. Jeśli operacje wykonywane przez Open WebUI należą do mniej wymagających, to może to prowadzić do potencjalnie niższego zużycia energii, co w konsekwencji może przekładać się na mniejszą moc obliczeniową dostępną dla testowanego modelu.

c. Wnioski z przeprowadzonych analiz

Analizując dane przedstawione dotychczas można dojść do wniosku, że istotnymi parametrami modeli może być szybkość generowania tokenów oraz zużycie mocy. Ważne jest również założenie dotyczące akceptowalnego progu szybkości odpowiedzi. Na rysunku 13 przedstawiono rozkłady szybkości generowania tokenów, uzyskiwane w trakcie badania różnych modeli LLM w środowisku Ollama. Można zauważyć, że średnia liczba tokenów generowanych w ciągu jednej mieściła się pomiędzy wartościami 1,84 (dla modelu Llama 3.1), a 7,71 (dla modelu Llama 3.2). Dla modeli Llama 3.1 oraz Phi 3.5 Mini subiektywnie odczuwane poziomy szybkości generowania odpowiedzi były zbyt niskie (odpowiednio 1,84 oraz 2,56). Szybkości generowania tokenów przez pozostałe modele były wystarczające. W związku z tym w badaniu założono minimalną wartość progową na poziomie 3,5 tokenów na sekundę. Na rys. 13 widać, że modele Gemma 2 w wersji z 2 miliardami parametrów firmy Google oraz Llama 3.1 z miliardem parametrów firmy Meta miały szybkości generowania tokenów większe od założonej wartości progowej.

Mimo iż nie wszystkie modele generują tokeny z szybkością większą niż założona wartość progowa, to nie można ich dyskwalifikować, jako modeli nie nadających się do wykorzystania. W przypadku zastosowań nie wymagających natychmiastowych

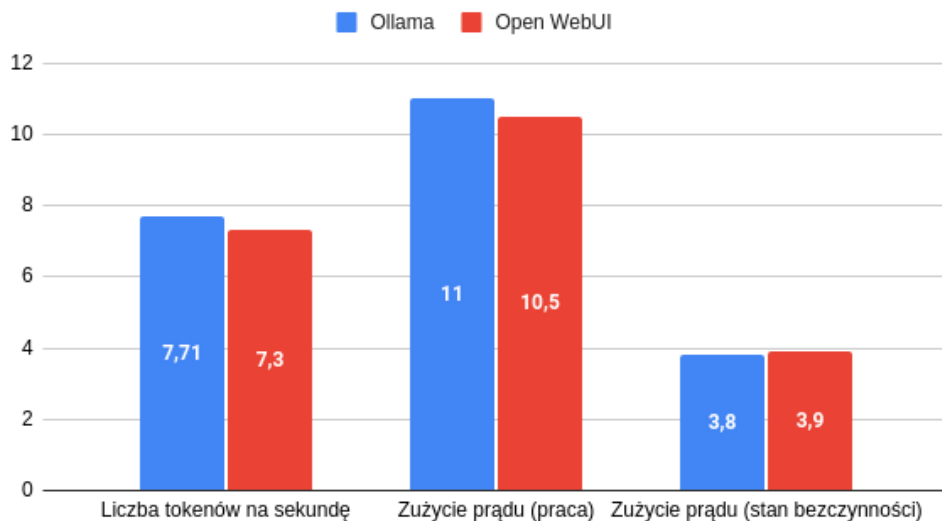
odpowiedzi, ale np. ceniących sobie dokładność, modele te mogą zostać z powodzeniem wykorzystywane.



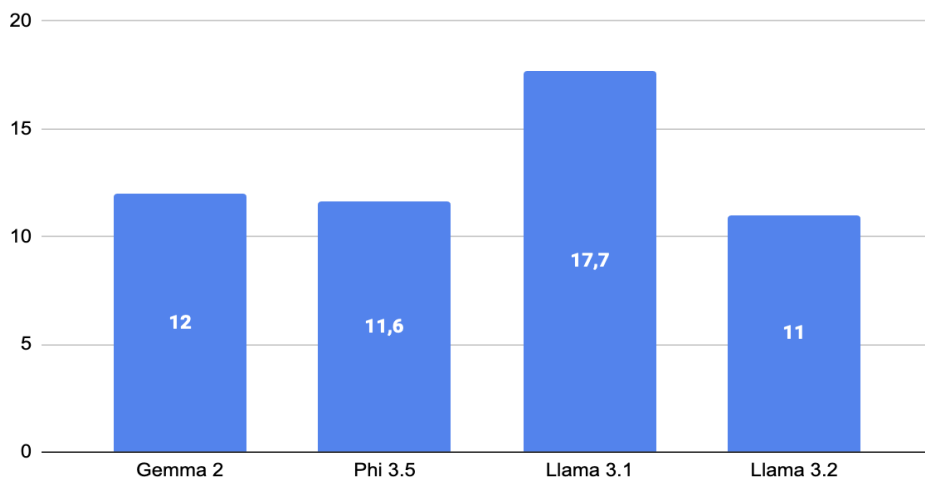
Rysunek 13. Średnia liczba generowanych tokenów na sekundę w środowisku Ollama

Istotne może być również przeanalizowanie wymagań odnoszących się do poziomu niezbędnych zasobów, wykorzystywanych przez modele sztucznej inteligencji na komputerze Raspberry Pi. Jedną z obaw dotyczących możliwości uruchomienia modeli na tym komputerze, była wielkość dostępnej pamięci operacyjnej. Okazało się jednak, że pamięć operacyjna nie była największym problemem napotkanym podczas analizy. Takim problemem okazała się wydajność procesora. Wydajność ta jest wystarczająca do tego, aby można było z powodzeniem użytkować mniejsze modele LLM. Jednakże uruchamiając dodatkowe aplikacje w tle, takie jak np. program Open WebUI, można zauważyć spadek wydajności modeli. W analizowanym przypadku, wydajność modelu Llama 3.2 w wersji z miliardem parametrów współpracującego z interfejsem graficznym Open WebUI była mniejsza niż wydajność tego modelu współpracującego wyłącznie ze środowiskiem Ollama. Zaobserwowano, że spadek wydajności modelu był nieznaczny i wynosił tylko około 6%. Zmieniło się również zużycie mocy przez urządzenie. Zapotrzebowanie na moc się zmniejszyło w stosunku do zapotrzebowania występującego podczas korzystania z modelu za pomocą samego oprogramowania Ollama (rys. 14).

Ollama i Open WebUI



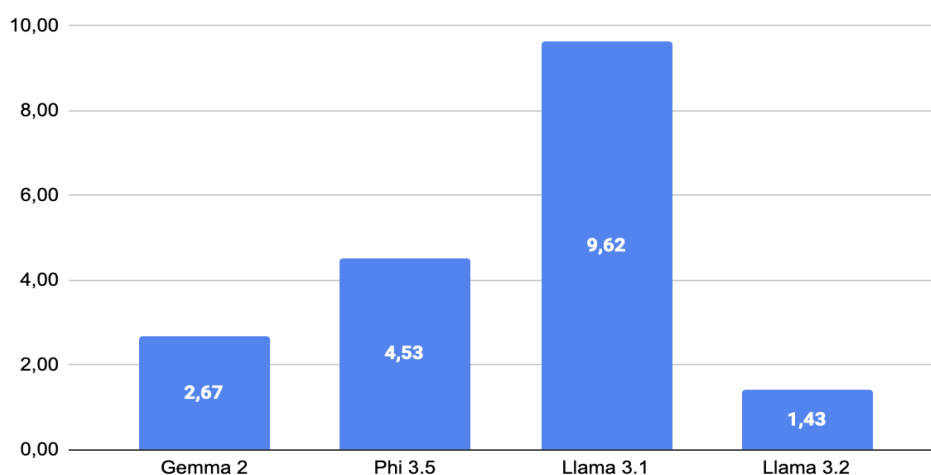
Rysunek 14. Porównanie szybkości generowania tokenów oraz zużycia mocy w różnych środowiskach pracy (Ollama, Open WebUI) dla modelu Llama 3.2



Rysunek 15. Średnie zużycie mocy podczas korzystania z wybranych modeli LLM w środowisku Ollama

Z drugiej strony, zapotrzebowanie na moc zmieniało się także wraz ze zmianą wykorzystywanych modeli. Moc wykorzystywana przez komputer Raspberry Pi była różna w trakcie pracy różnych modeli. Najwięcej mocy potrzebował model Llama 3.1, który był także największym badanym modelem. Pozostałe modele, mimo różnej liczby parametrów i wynikającego z tego zróżnicowanego zapotrzebowania na zasoby obliczeniowe, nie różniły się od siebie znacząco. Ich średnie zapotrzebowanie na moc różniło się maksymalnie o jeden wat.

Można zatem założyć, że modele poniżej 8 miliardów parametrów wykazują zużycie na poziomie 12 W lub mniej, sprawiając wrażenie bycia bardziej efektywnymi względem wykorzystywanej energii. Można to zwizualizować, określając, ile watów wybrany model potrzebuje do wygenerowania jednego tokena (rys. 16) Na podstawie przedstawionych na rys. 16 wyników widać, że mniejsze modele pozostają efektywniejsze energetycznie. Wymagania energetyczne wydają się być uzależnione od liczby parametrów. Niestety, dane te trudno jest porównywać ze względu na zapotrzebowanie energetyczne procesora, gdyż producent takich danych nie udostępnia.



Rysunek 16. Średnie zużycie mocy potrzebne do wygenerowania jednego tokena przez wybrane modele LLM pracujące w środowisku Ollama

Na podstawie zużycia mocy można także wyliczyć koszt związany z użytkowaniem. Jeśli założyć korzystanie jedynie z modelu Gemma 2, to przy skrajnym scenariuszu, w którym model ten pracuje cały czas, Raspberry Pi po 30 dniach zużyje około 8,64

kWh energii elektrycznej. Według cennika firmy PGE na okres od 1 listopada do końca 2024 roku, przy taryfie G11 jest to koszt, który nie przekroczy kwoty 5 zł i 57 groszy [23].

5. Podsumowanie

Za pośrednictwem Internetu można korzystać z coraz większej liczby modeli sztucznej inteligencji. W związku z tym, pojawiły się wątpliwości co do balansu, który z jednej strony obejmuje korzyści płynące z korzystania z takich modeli, a z drugiej strony uwzględnia pojawiające się zagrożenia i koszty. Zauważono, że podczas korzystania z rozwiązań firm trzecich, znaczącym zagrożeniem może być rezygnacja z prywatności. Z kolei, możliwość uruchomienia wybranych modeli na własnym urządzeniu dałaby większą kontrolę nad ruchem sieciowym. To pozwoliłoby na blokowanie niechcianych połączeń, a więc przyczyniłoby się do zwiększenia poziomu prywatności podczas konwersacji z wybranym modelem. Aby to sprawdzić, przeprowadzono eksperyment, w którym modele LLM implementowano na komputerze Raspberry Pi, badając jak modele te radzą sobie w zastosowaniach domowych. Niniejszy artykuł stanowi prezentację wyników uzyskanych w trakcie przeprowadzonego eksperymentu.

Platformę Raspberry Pi wybrano nieprzypadkowo. Jest ona znana między innymi z niskiego poboru mocy, co może skutecznie zredukować rachunki za prąd podczas korzystania z tego urządzenia. Na komputerze Raspberry Pi przeprowadzono instalację systemu operacyjnego, a następnie zbadano proces instalacji wybranego, odpowiedniego do tego celu oprogramowania. Umożliwiło to określenie wydajności analizowanych modeli, poprzez sprawdzenie szybkości generowania odpowiedzi oraz przetwarzania pytań. Dodatkowo zweryfikowano wybrane narzędzie, będące interfejsem graficznym, pozwalające na komunikację z wybranymi modelami. Pozwoliło to na zbadanie wydajności wybranego modelu. W celu zweryfikowania potencjalnych kosztów, zmierzono również zużycie mocy przez tytułowe urządzenie. Wykorzystano do tego watomierz, który umożliwiał podgląd zużycia mocy w trakcie pracy komputera Raspberry Pi. Porównano analizowane narzędzia pod kątem wydajności oraz poziom zużywanej mocy. Zweryfikowano wpływ dodatkowego oprogramowania na działanie wybranego modelu oraz omówiono teoretyczny koszt

działania, z wykorzystaniem informacji na temat aktualnych średnich cen energii elektrycznej, obowiązujących w chwili pisania niniejszego artykułu.

Za końcowy wniosek z przeprowadzonej analizy można uznać, że modele LLM mogą funkcjonować na komputerze Raspberry Pi w sposób komfortowy i wystarczająco szybki, pod warunkiem, że model zostanie dopasowany do rozwiązywanego problemu. Jeśli wymogiem jest szybkość działania, to najlepszym wyborem będą modele z mniejszą liczbą parametrów, a więc takie, które zawierają nie więcej niż 3 miliardy parametrów. Rozwiązania posiadające 8 miliardów parametrów będą potrzebowały znacznie więcej czasu i energii, aby wygenerować odpowiedź.

Po przeprowadzonej analizie można stwierdzić, że niewątpliwą zaletą mniejszych modeli LLM jest to, że można je z powodzeniem wykorzystywać w wielu różnych zastosowaniach. W przypadku osadzenia modelu LLM na urządzeniu klasy Raspberry Pi, może to oznaczać dostęp do szeregu nowych możliwości bez konieczności łączenia z Internetem, mimo tego, że dotychczas takie połączenie było koniecznością.

Istotna jest także możliwość usprawnienia wydajności urządzenia poprzez dodanie odpowiedniej karty rozszerzeń. Jedną z przykładowych potencjalnych propozycji może być wspomniany w p. 2.2 moduł wspierający obliczenia związane z generatywnymi modelami sztucznej inteligencji, takimi jak LLM. Może się to okazać szczególnie ważne w projektach, które wymagają korzystania z uruchamianych na urządzeniu Raspberry Pi dokładniejszych modeli, także dlatego, że to właśnie procesor tego komputera był wąskim gardłem obliczeniowych dla analizowanych rozwiązań.

Literatura

- [1] *ChatGPT*. [Online]. <https://chatgpt.com/>.
- [2] OpenAI. *ChatGPT Pricing*. [Online]. <https://openai.com/chatgpt/pricing/>.
- [3] K. Krzykowski, *Analiza wydajności wybranych modeli językowych na komputerze Raspberry Pi 5*, Warszawska Wyższa Szkoła Informatyki, Warszawa, Praca magisterska 2025.
- [4] *Gigabyte*. [Online]. <https://en.wikipedia.org/wiki/Gigabyte>.
- [5] *Gigabajt*. [Online]. <https://pl.wikipedia.org/wiki/Gigabajt>.

- [6] C. Dilmegani. *What is Large Multimodal Models (LMMs)?* [Online]. <https://research.aimultiple.com/large-multimodal-models/>.
- [7] Nvidia Corporation. *Large Language Models*. [Online]. <https://www.nvidia.com/en-us/glossary/large-language-models/>.
- [8] OVHcloud. *Co to jest duży model językowy (LLM)?* [Online]. <https://www.ovhcloud.com/pl/learn/what-is-large-language-model>.
- [9] F. Kurp, *Sztuczna inteligencja od podstaw*. Gliwice: Helion, 2023.
- [10] Raspberry Pi Ltd. *Raspberry Pi 5 product brief*. [Online]. <https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf>.
- [11] Raspberry Pi Ltd. *Raspberry Pi 4 Tech Specs*. [Online]. <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>.
- [12] Raspberry Pi Ltd. *Raspberry Pi dokumentacja procesorów*. [Online]. <https://www.raspberrypi.com/documentation/computers/processors.html>.
- [13] Hailo Technologies Ltd. *Hailo*. [Online]. <https://hailo.ai/>.
- [14] Hailo Technologies Ltd. *Hailo-10H M.2 Generative AI Acceleration Module*. [Online]. <https://hailo.ai/products/ai-accelerators/hailo-10h-m-2-generative-ai-acceleration-module/>.
- [15] Ollama (Github). [Online]. <https://github.com/ollama/>.
- [16] Open WebUI (Github). [Online]. <https://github.com/open-webui/>.
- [17] *Tokeny*. [Online]. <https://learn.microsoft.com/pl-pl/dotnet/ai/conceptual/understanding-tokens>.
- [18] *htop – an interactive process viewer*. [Online]. <https://htop.dev/>.
- [19] Watomierz GreenBlue GB202. Instrukcja obsługi. [Online]. https://www.ted.net.pl/p/_/00/0019/001906/gb202-instrukcja.pdf.
- [20] B.M. Sánchez. *What the Heck are Tokens? Demystifying the Building Blocks of AI Language*. [Online]. <https://www.linkedin.com/pulse/what-heck-tokens-demystifying-building-blocks-ai-maneiro-s%C3%A1nchez-ojmvf>.
- [21] A. Kacprzak, *Prompt engineering i ChatGPT. Poradnik skutecznej komunikacji ze sztuczną inteligencją*. Gliwice: Helion, 2024.
- [22] *Ollama*. [Online]. <https://www.ollama.com>.
- [23] PGE Obrót S.A. z siedzibą w Rzeszowie. *Zestaw cen rozliczeniowych dla cenników taryfowych innych niż Taryfa G*. [Online].

https://www.gkpge.pl/content/download/fb648c1ca12cfeba172a22d2385c5ed9/file/zestaw-cen-dla-cennikow-g-nie-pure_072024_web.pdf.

Performance Analysis of Selected Language Models on Raspberry Pi

Abstract

This paper discusses the possibilities of implementing selected large language models (LLM) on the Raspberry Pi computer. The term “Large Language Models” (LLM) is characterized and the capabilities of Raspberry Pi computer version 5 are discussed. It presents the concept, implementation, and results of the performance analysis of LLM models. Metrics regarding token processing speed and operating time of Llama 3.2, Llama 3.1, Phi 3.5 Mini and Gemma 2 models were analyzed. The average power consumption during the use of selected models and the average power consumption to generate one token by selected models were examined. Final conclusions were formulated.

Keywords: *Raspberry Pi, Large Language Models, LLM, Llama, Phi 3.5 Mini, Gemma 2, Ollama, Open WebUI*