УДК: 004.8 https://doi.org/10.15407/jai2025.03.094

В. Є. Луц[1], О. І. Безверхий[2]
[1,2]Національний транспортний університет, Україна
вул. Омеляновича-Павленка, 1, Київ, 01010
[1]tibet.septim@gmail.com
[2]o_bezver@ukr.net
[1]https://orcid.org/0009-0001-2948-6935
[2]https://orcid.org/0000-0002-0834-6335

ПОРІВНЯЛЬНИЙ АНАЛІЗ МОДЕЛЕЙ ДЛЯ РОЗПІЗНАВАННЯ УКРАЇНСЬКОГО МОВЛЕННЯ WHSIPER ТА VOSK

Анотація. У цій статті представлено порівняльний аналіз двох підходів до автоматичного розпізнавання мовлення (ASR) — трансформерної фокусної архітектури Whisper та Kaldi-орієнтованого рішення Vosk – із фокусом на розпізнавання української мови. Дослідження побудовано на експериментах, виконаних за допомогою двох бенчмарків: faster-whisper на GPU та локальних моделей Vosk на CPU. Оцінка проводилася з використанням стандартних метрик: Word Error Rate (WER) для вимірювання точності та Real-Time Factor (RTF) і середнього часу обробки для оцінки швидкості. Перед порівнянням тексти нормалізовані (усунення пунктуації, приведення до нижнього регістру, усунення зайвих пробілів) для забезпечення коректності підрахунку WER. Результати свідчень, які при наявності апаратного прискорення Whisper демонструють нижчі значення WER і значно кращу пропускну здатність порівнянно з Vosk на CPU, особливо для моделей medium та large-v3. Натомість Vosk підтверджує свою конкурентоспроможність у сценаріях з обмеженими ресурсами: він споживає менше пам’яті, є стабільним і детермінованим у повторних прогонках, що робить його придатним для вбудованих та офлайн-рішень. У статті також обговорено обмеження дослідження, зокрема невеликий тестовий набір даних і залежність результатів від параметрів апаратного забезпечення та налаштування декодування. На основі отриманих висновків сформульовано практичні рекомендації щодо вибору архітектури: для сервісів, де доступний GPU та критична точність — використання Whisper середнього/великого розміру; для автономних систем з обмеженими ресурсами — Vosk. Подальші дослідження потребують розширених наборів даних, адаптації моделей під локальну лексику та повноцінного аналізу варіантів гібридних розгортань.

Ключові слова: штучний інтелект, інформаційні системи, розпізнавання мовлення, мовні моделі, українська мова, інформаційні технології, Vosk, Whisper, WER, RTF, CUDA.

V. Luts[1], O. Bezverkhyi[2]
[1,2]National Transport University, Ukraine
1, Omelyanovicha-Pavlenko St., Kyiv, 01010
[1]tibet.septim@gmail.com
[2]o_bezver@ukr.net
[1]https://orcid.org/0009-0001-2948-6935
[2]https://orcid.org/0000-0002-0834-6335

COMPARATIVE ANALYSIS OF UKRAINIAN SPEECH RECOGNITION MODELS WHSIPER AND VOSK

Abstract. This paper presents a comparative analysis of two approaches to automatic speech recognition (ASR) — the Whisper transformer architecture and the Kaldi-based Vosk solution — with a focus on Ukrainian language recognition. The study is based on experiments performed using two benchmarks: faster-whisper on GPU and local Vosk models on CPU. The evaluation was performed using standard metrics: Word Error Rate (WER) to measure accuracy and Real-Time Factor (RTF) and average processing time to assess performance. Before comparison, the texts were normalized (punctuation removal, conversion to lower case, removal of extra spaces) to ensure correct WER calculation. The results show that, when hardware accelerated, Whisper demonstrates lower WER values and significantly better throughput compared to Vosk on CPU, especially for the medium and large-v3 models. Instead, Vosk proves its competitiveness in resource-constrained scenarios: it consumes less memory, is stable and deterministic in repeated runs, which makes it suitable for embedded and offline solutions. The paper also discusses the limitations of the study, including the small test dataset and the dependence of the results on the hardware parameters and decoding settings. Based on the conclusions obtained, practical recommendations for choosing an architecture are formulated: for services where GPUs are available and critical accuracy is critical, Whisper of medium/large size is recommended; for

autonomous systems with limited resources, Vosk. Further research requires expanded datasets, adaptation of models to local vocabulary, and a full analysis of hybrid deployment options.

Keywords: artificial intelligence, information systems, speech recognition, language models, Ukrainian language, information technologies, Vosk, Whisper, WER, RTF, CUDA.

Вступ

Автоматичне розпізнавання мовлення (ASR) продовжує залишатися фундаментальною технологією для широкого кола застосувань, від голосових асистентів до систем транскрипції для науки. Зі зростанням обчислювальної потужності та доступністю великих моделей трансформаторного типу відбувається зсув до моделей, навчених на великомасштабних багатомовних наборах даних. OpenAI Whisper [1] є представником цього підходу та демонструє хороші результати багатьма мовами завдяки глибокому контекстному моделюванню [17]. На противагу цьому, Vosk, який базується на Kaldi, продовжує залишатися оптимальним вибором для автономних та ресурсонезалежних рішень, оскільки його моделі та алгоритми оптимізовані для виконання на центральному процесорі та мають незначну залежність від зовнішніх сервісів.

Для української мови, яка має складну морфологію та багатий словниковий запас, питання вибору моделі є особливо важливим: необхідно збалансувати вимоги до точності розпізнавання, затримки відповіді та апаратних обмежень замовника. Саме це і є метою цього дослідження — систематично порівняти моделі у режимах, які найчастіше зустрічаються на практиці: Whisper з апаратним прискоренням на графічному процесорі та Vosk [3] на центральному процесорі.

Постановка проблеми

Сучасні практичні завдання ASR [6] часто передбачають компроміси між точністю та продуктивністю. Якщо сервіс має обробляти аудіопотік у режимі реального часу, затримка та пропускна здатність стають критичними, тоді як для пакетної обробки максимальна точність є важливішою. Крім того, вимоги до інфраструктури суттєво впливають на вибір технології. Для української мови очевидні питання стосуються того, наскільки велика трансформерна модель на базі графічного процесора виправдовує свої витрати на обладнання порівняно з більш економічними варіантами Kaldi [2] на базі центрального процесора.

Більш детально проблемні питання формулюються наступним чином:

– по-перше, яка різниця в показниках точності (наприклад, WER) між Whisper та Vosk для української мови за типових умов запису;

– по-друге, як порівнюються часові витрати (час розпізнавання на одиницю аудіо) при запуску Whisper на графічному процесорі та Vosk на центральному процесорі;

– по-третє, які практичні обмеження та ризики при застосуванні кожного підходу у виробничому середовищі.

Аналіз останніх досліджень та публікацій

Огляд літератури та відкритих джерел показує чітку тенденцію: великі трансформерні моделі, навчені на багатомовних наборах даних, демонструють покращену здатність до узагальнення в завданнях розпізнавання мовлення, особливо коли йдеться про мовні контексти та довгі залежності. Whisper [1], одна з таких моделей, показує конкурентні результати завдяки поєднанню масштабного навчання та трансформеної архітектури. Водночас, дослідження в галузі мовної інженерії та розробки ASR підкреслюють, що точність залежить не лише від архітектури, але й від якості та репрезентативності навчального dataset для певної мови чи діалекту.

Дослідження на основі Kaldi (Vosk) доводять ефективність класичних методів у випадках обмежених ресурсів та необхідності детермінізму. Архітектури Kaldi [2] дозволяють налаштовувати акустичні моделі та лексичні словники, що

дає переваги у вузькоспеціалізованих областях [19]. У сукупності сучасні праці рекомендують поєднувати підходи: наприклад, використовувати важкі трансформерні моделі для пакетної або хмарної обробки, а легкі рішення Kaldi - для вбудованих або автономних систем [2].

З практичної точки зору, важливо відзначити методологічні рекомендації щодо оцінки ASR [11]: правильна попередня обробка тексту (нормалізація, видалення розділових знаків, робота з числовими формами), репрезентативність тестових зразків (різні динаміки, різні акустичні умови), а також прозора звітність про конфігурації запуску моделі (версії бібліотек, параметри декодування, специфіка обладнання).

Мета дослідження

Метою цієї роботи є отримання порівнянних та інтерпретованих даних про продуктивність Whisper та Vosk для української мови в умовах, що імітують реальні сценарії використання: Whisper працює на графічному процесорі з використанням faster-whipser для забезпечення апаратного прискорення, а Vosk [1], [3] працює на центральному процесорі з локальними моделями. Мета дослідження - відповісти на питання щодо компромісу між WER/затримкою, визначити межі застосування кожного підходу та сформулювати рекомендації для практичного впровадження у випадках, коли апаратні обмеження або вимоги до реального часу є критичними.

Методологічний підхід

Порівняльний аналіз було реалізовано шляхом розробки двох бенчмарків, які підтримують однакову логіку навчання та оцінки результатів. Обидва скрипти записували кілька коротких зразків мовлення (зазвичай по 6 секунд кожен), після чого власноруч вводився текст для обчислення WER. Перед обчисленням похибки здійснювалася стандартизація тексту, яка передбачала перетворення всіх символів до нижнього регістру, вилучення пунктуаційних знаків та надлишкових пробілів [4]. Такий підхід забезпечував відтворюваність та дозволяв мінімізувати систематичні відмінності у обчисленні WER.

У випадку Whisper, для пришвидшення операцій зниження точності було використано середовище з доступом до CUDA-сумісного графічного процесора [13] та використанням compute_type=float16, що значно скорочує час оцінки без значної втрати якості транскрипції [8]. Для Vosk тестування проводилося на центральному процесорі, що відображає реальне використання бібліотеки в автономних умовах. Обидва набори експериментів генерували зведені звіти та файли графіків у спільному каталозі, що дозволило провести порівняльний аналіз отриманих метрик.

Опис вимірювань та критеріїв

Основними метриками, що використовувалися в дослідженні, були середній час розпізнавання на зразок (латентність) та середній WER по набору зразків. Час вимірювався з моменту передачі аудіо до інтерфейсу моделі до отримання остаточної транскрипції [7]. WER розраховувався після нормалізації, що виключало відмінності в пунктуації та незначні відмінності у форматуванні. WER обчислюється як відношення кількості помилок до загальної кількості слів в еталонному тексті [12]:

$$WER = (S + D + I) / N * 100\%$$

де:

S — кількість замін (substitutions), коли слово розпізнано іншим словом;

D — кількість пропусків (deletions), коли слово не було розпізнано;

I — кількість вставок (insertions), коли модель додала зайві слова;

N — загальна кількість слів у еталонному (правильному) тексті.

У варіанті бенчмарків швидкість виражають як середній час обробки одного аудіофайлу або як реальне відношення часу обробки до тривалості аудіо (Real-Time Factor, RTF):

$$\mathbf{AVG\_TIME = (\Sigma t_i) / M}$$

де:

t_i — час розпізнавання і-го зразка,

M — кількість зразків.

RTF = T_обробки / T_аудіо

де:

T_обробки — загальний час, витрачений моделлю на транскрипцію,

T_аудіо — сумарна тривалість оброблених аудіофайлів.

RTF < 1 означає, що модель працює швидше за реальний час;

RTF > 1 — модель працює повільніше за реальний час;

Такий вибір метрик дозволяє чітко порівнювати продуктивність як у режимі реального часу, так і в контексті пакетної обробки.

Основні результати та їх інтерпретація

Результати, отримані під час типових бенчмарків, показують узагальнену картину, яка відповідає очікуванням щодо архітектурних переваг: Whisper, що працює на GPU, демонструє нижчий середній WER порівняно з Vosk на CPU (див. Рис. 1, 2), особливо коли йдеться про моделі середнього та великого розміру (medium, large-v3) [16]. Це пояснюється більшою потужністю моделі та її здатністю фіксувати контекстні залежності [5], [14], [15], що критично важливо для української мови з її варіативністю форм.

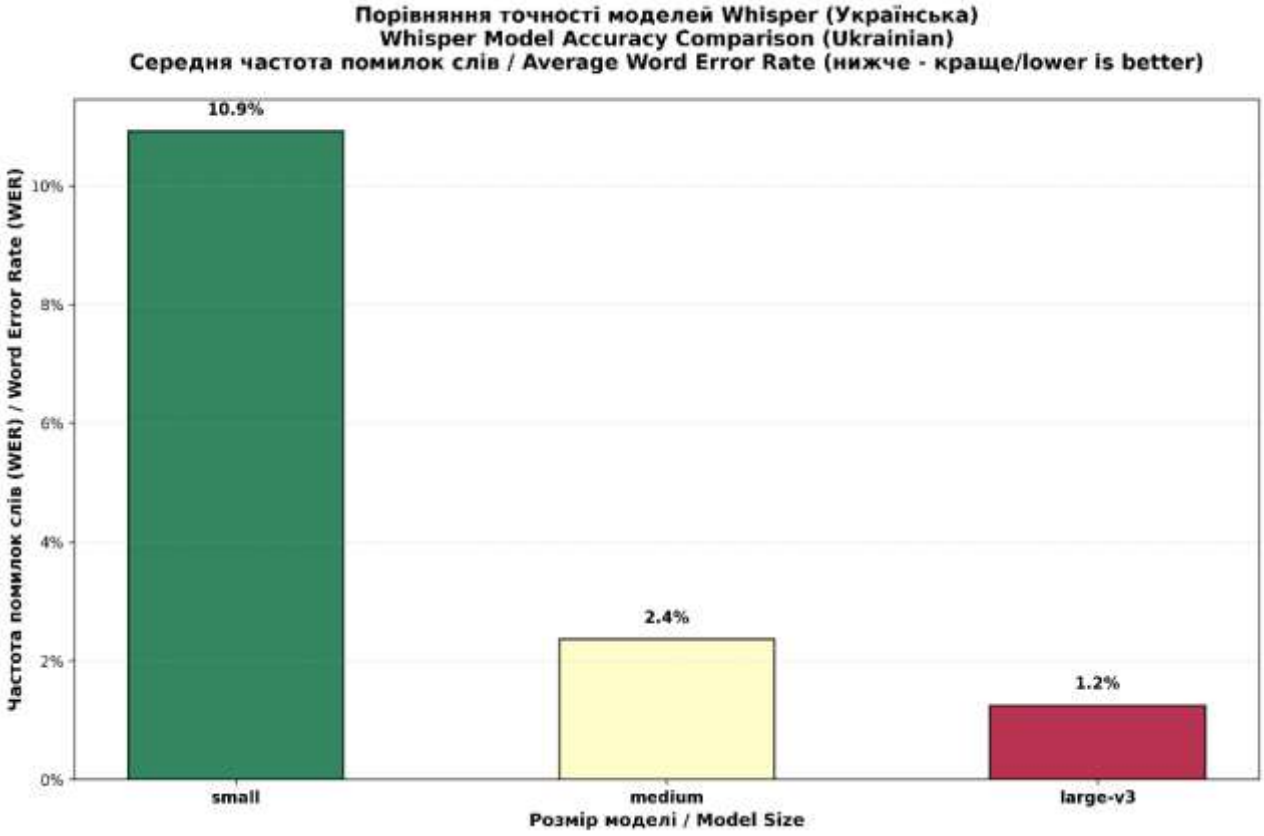

Рис. 1. Графік порівняння моделей Whsiper (WER)

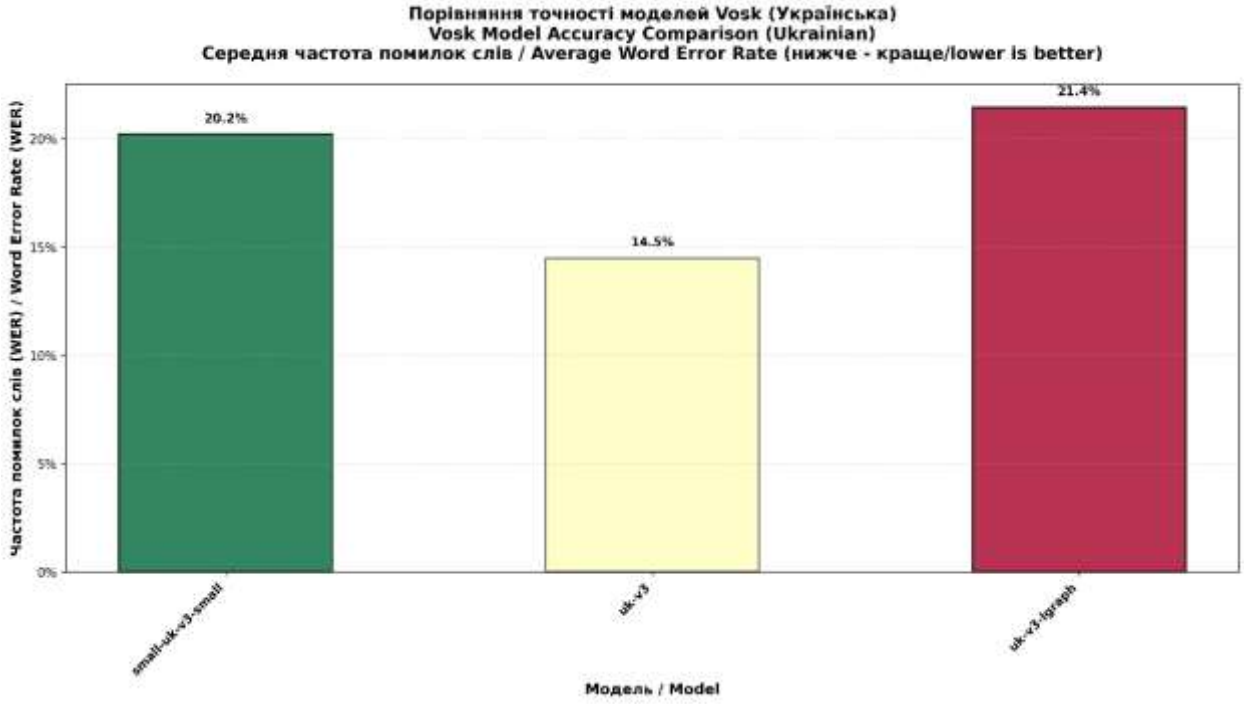

Рис. 2. Графік порівняння моделей Vosk (WER)

Щодо швидкості, використання GPU забезпечує явну перевагу: затримка розпізнавання для Whisper на конкурентному GPU може бути значно меншою, ніж для Vosk на поточному поколінні CPU, особливо у випадку пакетної обробки або потокового паралельного розпізнавання. Однак слід зазначити, що для дуже малих моделей Whisper (tiny або base) різниця в часі порівняно з Vosk може бути незначною, а в деяких конфігураціях перевага може перейти на користь Vosk завдяки його простішій архітектурі та меншим витратам на ініціалізацію.

Крім того, дослідження показало, що Vosk є більш стійким з точки зору детермінізму результатів при багаторазових запусках з однаковими налаштуваннями, тоді як трансформерні моделі можуть демонструвати невеликі варіації результатів залежно від випадковості в процесі декодування та тонкого налаштування гіперпараметрів (див. рис. 3, 4). Це важливо для застосувань, де важлива повна відтворюваність результатів.

Практичні наслідки

Коли система має доступ до потужного графічного процесора, а завдання вимагає відносно високої точності та низької затримки, інвестиції у велике розгортання Whisper виправдані – воно забезпечує кращу якість транскрипції та менший час відгуку. Для сценаріїв, де інфраструктура обмежена або де автономія та мінімальна залежність від сторонніх компонентів є пріоритетом, оптимізовані моделі Vosk залишаються привабливим вибором. Для багатьох практичних завдань оптимальним може бути гібридний підхід: використання легких моделей на кінцевих пристроях для попередньої обробки та фільтрації, а також більш ресурсоємних трансформаторів у хмарі для остаточної обробки та покращення якості [5], [20].

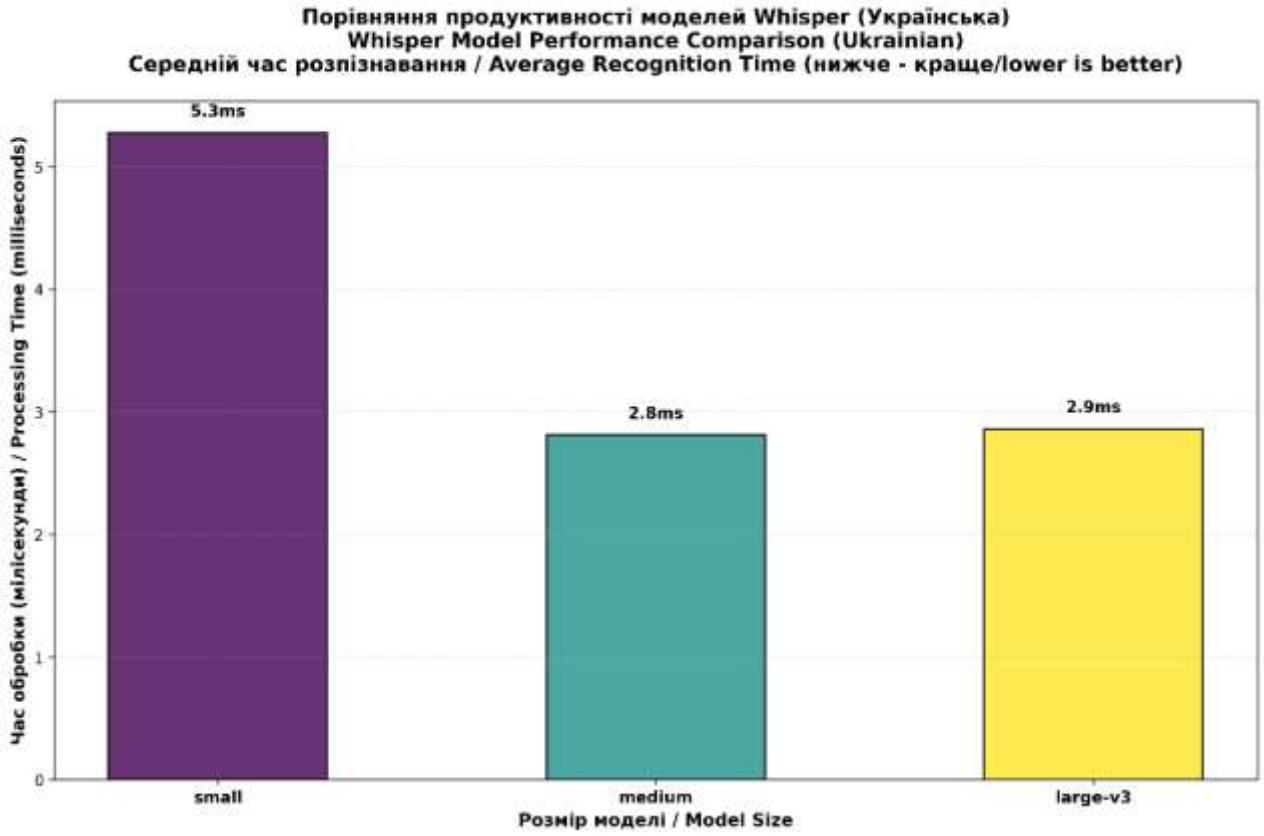

Рис. 3. Графік порівняння моделей Whisper (ART latency)

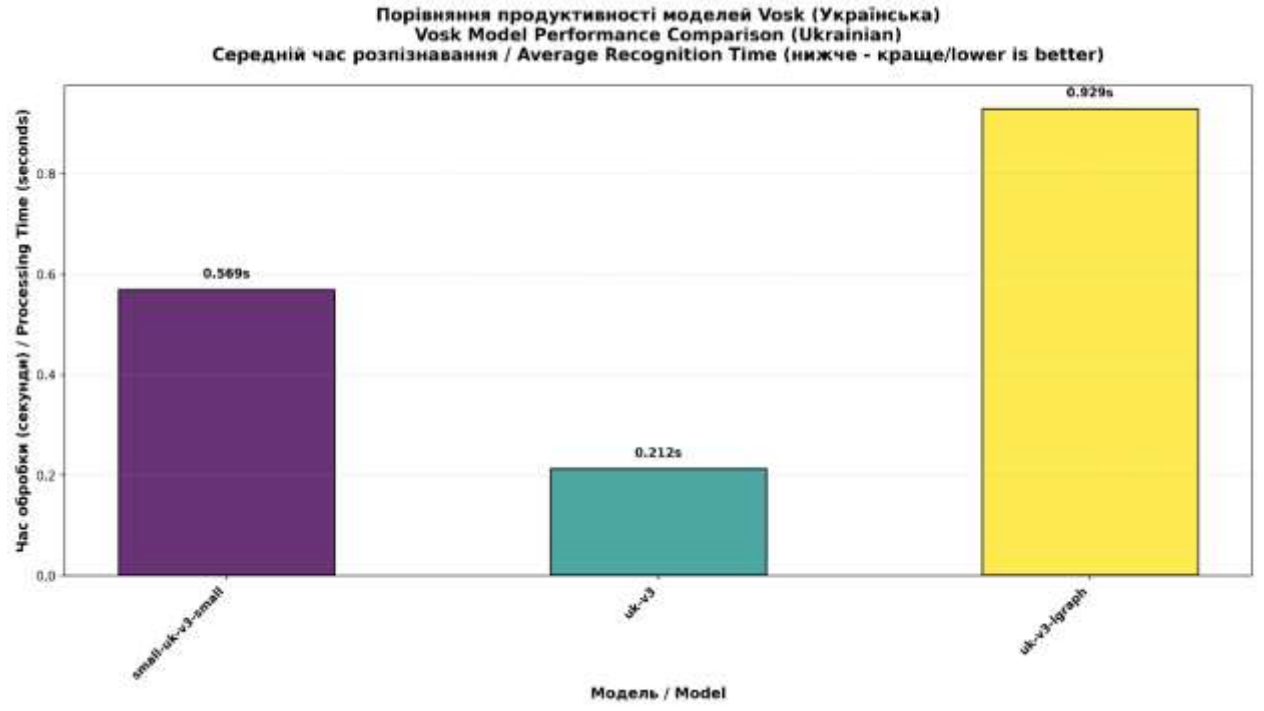

Рис. 4. Графік порівняння моделей Vosk (ART latency)

Обмеження дослідження

Слід зазначити, що проведене дослідження має обмеження, які впливають на узагальнюваність результатів. По-перше, обмежений розмір тестового набору (короткі вибірки тривалістю 6 секунд та 10 повторень) не дозволяє робити статистично обґрунтовані висновки для широкого діапазону мовних умов. По-друге, результати дуже

специфічні для апаратного забезпечення: тип і покоління графічного процесора, архітектура процесора, доступ до оперативної пам'яті та швидкість зберігання впливають на кінцеву затримку. По-третє, параметри декодування та деталі попередньої обробки аудіо виявилися важливими для об'єктивної оцінки, і невеликі зміни в цих налаштуваннях можуть суттєво змінити WER [12] (коефіцієнт відхилення від реального часу).

Тому для отримання більш узагальнюючих висновків необхідні подальші експерименти з більшими та різноманітнішими наборами даних, включаючи варіації фонового шуму, вікові групи мовників, діалектні відмінності та умови запису.

Висновки

Загалом, порівняльний аналіз показує, що сімейство трансформерних моделей Whisper перевершує якість транскрипції української мови при апаратному прискоренні. Whisper на основі графічного процесора забезпечує нижчі значення WER та може зменшити затримку до рівнів, придатних для завдань реального часу високої складності. Vosk на основі процесора залишається актуальним у сценаріях з обмеженими ресурсами або там, де пріоритетом є автономність, низький рівень пам'яті та поведінковий детермінізм.

У практичних рекомендаціях варто наголосити, що вибір між моделями має ґрунтуватися на конкретних вимогах проєкту, доступних ресурсах та бажаній архітектурі розгортання. У цьому випадку важливо враховувати не лише показники WER та затримки в лабораторних умовах, але й експлуатаційні аспекти: стабільність роботи під навантаженням, витрати на інфраструктуру, складність підтримки та оновлення моделей, а також питання безпеки та конфіденційності оброблюваних аудіопотоків.

У хмарних середовищах обробки та сервісах з високою частотою доступу, де критерієм успіху є одночасна стабільність відгуку та висока точність, використання Whisper у поєднанні з екземплярами GPU часто виправдане. У таких випадках економічна модель проєкту повинна враховувати вартість часу GPU та потенційне скорочення людських витрат на пост-редагування транскрипції, яка зазвичай вимагає меншої ручної корекції при використанні великих трансформаторних моделей. Однак важливо планувати робоче навантаження — наприклад, використовувати автомасштабування екземплярів або використовувати чергування пакетної та онлайн-обробки для зменшення пікових витрат.

У локальних застосуваннях, вбудованих пристроях та рішеннях з обмеженим доступом до мережі пріоритетом залишається автономність системи. Тут Vosk має низку переваг: мінімальна залежність від зовнішніх сервісів, низькі вимоги до пам'яті та стабільна поведінка під час багаторазових запускав. Для проєктів, що вимагають повної автономної роботи або роботи в середовищах з обмеженими ресурсами (наприклад, пристрої Інтернету речей або промислові контролери), Vosk дозволяє реалізувати прийнятну якість розпізнавання без інвестування в дорогі обчислювальні ресурси. Водночас, для підвищення точності в таких сценаріях корисні методи адаптації: тонке налаштування акустичних модулів, розширення словника специфічною термінологією та застосування постобробки результатів за допомогою граматичних або лексичних правил.

Для дослідницьких та наукових завдань, де потрібна висока відтворюваність експериментів та можливість детального аналізу помилок, доцільно комбінувати підходи. Початкове очищення та класифікація аудіопотоку можуть виконуватися за допомогою легких локальних моделей (Vosk або невеликих версій Whisper), після чого складніші сегменти або важливі фрагменти передаються на хмарну обробку за допомогою великих моделей Whisper. Така стратегія дозволяє оптимізувати витрати, зберігаючи при цьому високу якість

кінцевої транскрипції для критичних ділянок.

Технічні аспекти інтеграції вимагають особливої уваги до сумісності аудіоформатів, нормалізації сигналу та обробки шуму [9]. Практика показує, що погана підготовка аудіопотоку (непередбачувані піки гучності, нестабільна частота дискретизації, сильний фон) може значно погіршити результати навіть найкращих моделей. Тому варто додати до етапу модуля розпізнавання компоненти попередньої обробки: нормалізацію гучності, просте видалення шуму та перевірку цілісності файлу. Також корисно підтримувати метадані для кожного зразка для подальшого аналізу та вдосконалення системи.

Ще одним важливим аспектом є практичність експлуатації: процеси оновлення моделей, моніторинг якості та механізми аварійного відновлення [10]. Продакшн-системи повинні мати налагоджені канали для відстеження погіршення якості (наприклад, періодичні перевірки WER на контрольній підмножині записів) та можливість швидкого відкату до стабільніших версій моделей у разі виникнення проблем. Для Whisper це може означати керування версіями моделей та контроль використання ресурсів GPU, для Vosk — процеси переналаштування мовних словників та акустичних профілів у разі появи нових доменів або термінології.

У контексті подальших досліджень рекомендується проводити експерименти, що включають більші dataset, різні варіанти обробки шуму та адаптацію моделей до місцевих мовних особливостей. Крім того, доцільно дослідити економічну ефективність різних архітектур розгортання. Нарешті, для складних сценаріїв рекомендується оцінити можливість точного налаштування Whisper на специфічних для домену даних, а потім порівняти підвищення точності та витрати ресурсів такого підходу.

Література

1. Radford, A. (2022). Whisper: Robust speech recognition. OpenAI Technical Report. Отримано з https://cdn.openai.com/papers/whisper.pdf
2. Povey, D. (2011). The Kaldi speech recognition toolkit. Отримано з https://kaldi-asr.org/
3. Alphacephei. Vosk API documentation. Отримано з https://alphacephei.com/vosk/
4. Jurafsky, D., & Martin, J.H. (2020). Speech and language processing (3rd ed.). Prentice Hall. Отримано з https://web.stanford.edu/~jurafsky/slp3/
5. Han, K. (2020). ContextNet: Improving convolutional neural networks for ASR. Отримано з https://arxiv.org/abs/2005.03191
6. Park, D.S. (2019). SpecAugment: A simple data augmentation method for ASR. Отримано з https://arxiv.org/abs/1904.08779
7. Pratap, V. (2020). Wav2letter++: The fastest open-source speech recognition system. ICASSP. Отримано з https://arxiv.org/abs/1812.07625
8. Kuchayev, O. (2019). Mixed precision training for speech recognition. ICASSP. Отримано з https://doi.org/10.1109/ICASSP.2019.8682590
9. Tóth, L. (2015). Combining articulatory and acoustic information in DNN-based ASR. Computer Speech & Language. Отримано з https://doi.org/10.1016/j.csl.2015.06.001
10. Besacier, L.,(2014). Automatic speech recognition for under-resourced languages. Speech Communication. Отримано з https://doi.org/10.1016/j.specom.2014.01.002
11. Panayotov, V. (2015). Librispeech: An ASR corpus based on public domain audiobooks. ICASSP. Отримано з https://doi.org/10.1109/ICASSP.2015.7178964
12. Jiwer. Jiwer Python library documentation. Отримано з https://github.com/jitsi/jiwer
13. NVIDIA. (2023). CUDA toolkit documentation. Отримано з https://docs.nvidia.com/cuda/
14. Chan, W. (2016). Listen, attend and spell. ICASSP. Отримано з https://doi.org/10.1109/ICASSP.2016.7472621
15. Baevski, A. (2020). Wav2vec 2.0: A framework for self-supervised learning of speech representations. NeuroIPS. Отримано з https://arxiv.org/abs/2006.11477
16. Stolcke, A. (2017). Effects of language model size on speech recognition performance. ICASSP. Retrieved from https://doi.org/10.1109/ICASSP.2017.7952696
17. Williams, W. (2019). Contextual speech recognition in end-to-end neural models. Отримано з https://www.isca-speech.org/archive/Interspeech_2019/williams19_Interspeech.html

References

1. Radford, A. (2022). Whisper: Robust speech recognition. OpenAI Technical Report. Retrieved from https://cdn.openai.com/papers/whisper.pdf
2. Povey, D. (2011). The Kaldi speech recognition toolkit. IEEE ASRU. Retrieved from https://kaldi-asr.org/
3. Alphacephei. Vosk API documentation. Retrieved from https://alphacephei.com/vosk/
4. Jurafsky, D., & Martin, J.H. (2020). Speech and language processing (3rd ed.). Prentice Hall. Retrieved from https://web.stanford.edu/~jurafsky/slp3/
5. Han, K. (2020). ContextNet: Improving convolutional neural networks for ASR. Interspeech. Retrieved from https://arxiv.org/abs/2005.03191
6. Park, D.S. (2019). SpecAugment: A simple data augmentation method for ASR. Interspeech. Retrieved from https://arxiv.org/abs/1904.08779
7. Pratap, V., et al. (2020). Wav2letter++: The fastest open-source speech recognition system. ICASSP. Retrieved from https://arxiv.org/abs/1812.07625
8. Kuchayev, O., et al. (2019). Mixed precision training for speech recognition. ICASSP. Retrieved from https://doi.org/10.1109/ICASSP.2019.8682590
9. Tóth, L. (2015). Combining articulatory and acoustic information in DNN-based ASR. Computer Speech & Language. Retrieved from https://doi.org/10.1016/j.csl.2015.06.001
10. Besacier, L. (2014). Automatic speech recognition for under-resourced languages. Speech Communication. Retrieved from https://doi.org/10.1016/j.specom.2014.01.002
11. Panayotov, V. (2015). Librispeech: An ASR corpus based on public domain audiobooks. ICASSP. Retrieved from https://doi.org/10.1109/ICASSP.2015.7178964
12. Jiwer. Jiwer Python library documentation. Retrieved from https://github.com/jitsi/jiwer
13. NVIDIA. (2023). CUDA toolkit documentation. Retrieved from https://docs.nvidia.com/cuda/
14. Chan, W. (2016). Listen, attend and spell. ICASSP. Retrieved from https://doi.org/10.1109/ICASSP.2016.7472621
15. Baevski, A. (2020). Wav2vec 2.0: A framework for self-supervised learning of speech representations. NeuroIPS. Retrieved from https://arxiv.org/abs/2006.11477
16. Stolcke, A. (2017). Effects of language model size on speech recognition performance. ICASSP. Retrieved from https://doi.org/10.1109/ICASSP.2017.7952696
17. Williams, W. (2019). Contextual speech recognition in end-to-end neural models. Interspeech. Retrieved from https://www.isca-speech.org/archive/Interspeech_2019/williams19_Interspeech.html

The article has been sent to the editors 30.08.25.
After processing 13.09.25.
Submitted for printing 30.09.25.