Київський національний університет імені Тараса Шевченка
Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

ЖЕЛЕЗНЯКОВ ДМИТРО ВАЛЕНТИНОВИЧ

УДК 004.021:004.023:004.932

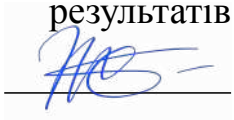
ДИСЕРТАЦІЯ

МЕТОДИ РОЗПІЗНАВАННЯ РУКОПИСНИХ МАТЕМАТИЧНИХ ВИРАЗІВ НА ОСНОВІ МАШИННОГО НАВЧАННЯ ТА КОНТЕКСТНО-ВІЛЬНИХ ГРАМАТИК В УМОВАХ ОБМЕЖЕНЬ

122 — Комп’ютерні науки

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело



_____ Д.В. Железняков

Науковий керівник

Терещенко Василь Миколайович,
доктор фізико-математичних наук, професор

Київ – 2022

АНОТАЦІЯ

Железняков Д.В. Методи розпізнавання рукописних математичних виразів на основі машинного навчання та контекстно-вільних граматик в умовах обмежень. Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки» – Київський національний університет імені Тараса Шевченка МОН України, Київ, 2022.

Мова математичних виразів використовується науковцями, інженерами, студентами та іншими спеціалістами у багатьох галузях. Це універсальна мова, що застосовується і яку розуміють у всьому світі. Проте створення цифрових документів, що містять складні математичні вирази, значно складніше, ніж створення документів, що містять простий текст. Це пов'язано зі складнішою двовимірною структурою математичних виразів та великою кількістю математичних символів. Зазвичай ввід математичних виразів вимагає від користувача знань спеціальних математичних нотацій або застосування складних редакторів із величезною кількістю елементів та складною навігацією. Однак найбільш природним інтерфейсом для вводу такої інформації є ручка або перо/стилус. При цьому інтерфейс із пером підтримує не тільки ввід початкової інформації, але й забезпечує можливість вносити зміни до вже існуючих електронних документів таким же природним чином. До недавнього часу електронною ручкою були оснащені переважно мобільні пристрої, такі як смартфони та планшети. Сьогодні нові типи пристроїв, зокрема інтерактивні панелі та розумні поверхні для письма, набули поширення в офісах та навчальних закладах, відкриваючи нові можливості для технологій розпізнавання певного рукописного вмісту, такого як математика, схеми, діаграми, таблиці, ескізи тощо.

Розпізнавання рукописних математичних виразів відноситься до задач розпізнавання образів і є важливою частиною розпізнавання та аналізу документів. Традиційно виділяють декілька етапів розпізнавання математичних виразів, а саме сегментація символів, класифікація символів та структурний аналіз. З розвитком наскрізних рішень всі ці етапи були об'єднані в рамках однієї нейронної мережі. Однак наскрізні рішення є дуже ресурсомісткими і на даний момент не забезпечують можливості редагування виразів за допомогою рукописних жестів та символів.

Метою цієї роботи є створення нового методу для онлайн розпізнавання рукописних математичних виразів на пристроях, що мають обчислювальні обмеження. Головна ідея полягає в поєднанні можливостей рекурентних нейронних мереж та граматичних методів розбору для забезпечення розпізнавання рукописних математичних виразів на мобільних пристроях з високим рівнем якості. Також рішення має забезпечувати можливість послідовного вводу математичних виразів, де час відгуку системи є критично важливим.

Розроблена система розпізнавання рукописних математичних виразів складається з двох етапів. На першому етапі виконується сегментація та класифікація символів за допомогою нейронних мереж, на другому будується найбільш ймовірне дерево математичного виразу.

Сегментація та класифікація символів виконуються за допомогою архітектури нейронних мереж, що має назву двонаправлена довга короткочасна пам'ять. Навчання цієї нейронної мережі побудоване з використанням нейромережевої часової класифікації як цільової функції втрат. Завдяки можливості запам'ятовувати довгострокові залежності така комбінація успішно використовується в багатьох задачах розпізнавання одновимірних мов, де правильна сегментація не вимагається. У цій роботі така комбінація технологій адаптована для одночасного вирішення двох окремих задач, а саме сегментації та класифікації символів. У розпізнаванні математичних виразів дуже важлива не тільки правильна класифікація символів, а й правильна сегментація, оскільки результат сегментації впливає на подальші етапи. З метою покращення якості сегментації та класифікації були

запропоновані додаткова легковагова нейронна мережа для класифікації окремих символів та алгоритм інтеграції результатів двох нейронних мереж. Цей алгоритм забезпечує уточнення результатів класифікації з урахуванням можливих варіантів сегментації символів. Інтеграція додаткової нейронної мережі дала можливість зменшити помилки у розпізнаванні всього виразу приблизно на 10%.

Побудова дерева математичного виразу здійснюється за допомогою двовимірної стохастичної контекстно-вільної граматики та алгоритму розбору. Метод розбору базується на алгоритмі Кокке-Янгера-Касамі та адаптований під двовимірні мови. Для вирішення задачі класифікації просторових відношень між елементами математичного виразу були запропоновані набір геометричних ознак та покращена концепція «body box» за рахунок локального контексту. На базі порівняння точності та швидкості різних алгоритмів класифікації просторових відношень було запропоновано використання дерева ухвалення рішень, оскільки цей алгоритм більшою мірою відповідає вимогам щодо обмеження обчислювальних ресурсів.

Оцінка правдоподібності математичного виразу здійснюється за допомогою інтеграційної функції, що поєднує результати окремих класифікаторів: сегментації та класифікації символів, просторових відношень, мовної та граматичної моделей. Граматичні методи розбору є вимогливими до обчислювальних ресурсів та можуть мати факторіальну часову складність. У цій роботі для зменшення часової складності алгоритму розбору був запропонований набір методів, що спираються на методи обчислювальної геометрії, теорії графів та динамічного програмування. Інтеграція променевого пошуку з динамічним регулюванням ширини променя дає можливість уникнути недоліків жадібних алгоритмів. Побудова імен функцій забезпечує зменшення розмірності вхідних даних. Підхід, заснований на регіоні пошуку, забезпечує мінімізацію складності за рахунок геометричного пошуку потрібних елементів у побудові дерева математичного виразу. Дерево домінування забезпечує зменшення складності алгоритму з вертикально розташованими елементами. Регіони охоплення запобігають створенню непотрібних проміжних вузлів у побудові дерева розбору.

Однією з проблем розробки систем із нейронними мережами є необхідність у наборах даних великого розміру. Задача підготовки наборів даних для розпізнавання рукописних математичних прикладів ускладнюється потребою детальної анотації цих наборів. Анотація математичних виразів має ієрархічну структуру та складається з декількох рівнів. Кожен рівень анотації використовується для тренування різних моделей та оцінки відповідних етапів системи розпізнавання. В цій роботі запропонований новий метод для напівавтоматичної анотації набору рукописних математичних виразів, який не вимагає початкової перевірки зразків або попередньо підготовлених моделей. Метод продемонстрував високу точність анотації, яка склала близько 99% на рівні виразів, та дав можливість анотувати понад 85% усіх зразків. Запропонований метод допоміг зменшити обсяг ручної роботи більш ніж на 90%. Також цей метод може бути застосований до наборів даних з високим рівнем шуму та дозволяє виявляти зразки, які вимагають ручної перевірки.

Оцінка точності розпізнавання здійснювалася на відкритих наборах даних CROHME, а оцінка швидкодії – на мобільних пристроях. Для порівняння з іншими системами використовувалася метрика точності розпізнавання усього виразу, яка є дуже чутлива до будь-яких помилок. Так розроблений метод демонструє високі показники якості. Точність розпізнавання виразу для алфавіту розміром 101 символ становить 73.12%, 71.75% та 72.31% на CROHME 2014, 2016, 2019 відповідно. Оцінка ефективності методів зменшення обчислювальної складності була здійснена в серії експериментів із застосуванням методу абляцій. Запропоновані методи дали змогу зменшити час виконання структурного аналізу з 5–7 секунд до 20–40 мілісекунд. У результаті середній час розпізнавання виразу на мобільному пристрої становить близько 55 мілісекунд.

Система, яка базується на методах, запропонованих у цій роботі, брала участь у відкритому змаганні з розпізнавання рукописних математичних виразів CROHME у 2019 році. Запропонований варіант системи посів друге місце з точності розпізнавання виразів на рівні 79.82%. Також система була адаптована для

змагання з офлайн розпізнавання та виявлення рукописних математичних виразів у 2020 році та показала якість розпізнавання на рівні 61.90%.

Розроблену систему було успішно застосовано в декількох мобільних додатках, що засновані на інтерфейсі з пером. Представлені додатки дають можливість вводити складні документи в більш швидкий та природний спосіб. Рукописний калькулятор для вводу, редагування та обчислення математичних виразів демонструє здатність інтегрувати запропонований метод для побудови ітеративного вводу математичних виразів. На основі додатка «Інтерактивний папір» продемонстровано приклад інтелектуального інтерфейсу з пером для вводу складних документів, які містять текст, математичні вирази, таблиці, графіки та діаграми. Було досліджено зручність користування інтерфейсу з пером шляхом виконання завдань відносно великою кількістю учасників. Представлений у цій роботі метод розпізнавання рукописних математичних виразів був впроваджений у комерційний додаток «S Note» для *Samsung Galaxy Note 9* у вигляді окремого плагіну для трансформації набору рукописних штрихів у друковане представлення математичного виразу.

Робота є складовою частиною наукових досліджень, які ведуться на кафедрі математичної інформатики факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка.

Ключові слова: рукописний математичний вираз, онлайн розпізнавання, рекурентна нейронна мережа, контекстно-вільна граматика, людино-комп'ютерна взаємодія, орієнтований на перо інтерфейс, всюдисущі та мобільні обчислення

ANNOTATION

Zhelezniakov D.V. Methods for handwritten mathematical expression recognition based on machine learning and context-free grammars under constraints. Qualification scientific work in the form of manuscript.

Dissertation thesis for acquiring the degree of a Doctor of Philosophy in speciality 122 «Computer science.» – Taras Shevchenko National University of Kyiv, Kyiv, 2022.

The language of mathematical expressions is an essential part of many domains, including education, engineering, and science. It is a universal language used and understood all over the world. However, creating digital documents that contain complex mathematical expressions is considerably more difficult than creating documents that contain simple text. This is due to the more complex two-dimensional structure of mathematical expressions and a large of mathematical symbols alphabet. Usually, mathematical expressions input requires the user to know special mathematical notations or complex editors with many elements and tangled navigation. However, a pen or stylus is the most natural interface for inputting such information. At the same time, the pen-based interface supports the input of the initial information and provides an ability to change existing electronic documents in the same natural way. Until recently, mainly mobile devices such as smartphones and tablets were equipped with an electronic pen. Nowadays, new types of devices such as interactive panels, digital pens, and smart writing surfaces have become widely adopted in offices and educational institutions, opening up new opportunities for technologies for recognizing specific handwritten content such as mathematics, diagrams, charts, tables, sketches, etc.

Handwritten mathematical expressions recognition is one of the pattern recognition tasks and is a significant part of document recognition and analysis. Recognition of mathematical notation traditionally involves several stages such as character segmentation, character classification, and structural analysis. With the development of

end-to-end solutions, all these stages have been combined into a single neural network. However, end-to-end solutions are very resource-intensive and do not support editing expressions using handwritten gestures and symbols.

This work aims to create a new method for online recognition of handwritten mathematical expressions for devices that have computational limitations. The main idea of this work is to combine the capabilities of recurrent neural networks and grammatical parsing methods to ensure the recognition of handwritten mathematical expressions on mobile devices with a high level of accuracy. The solution should also support the sequential input of mathematical expressions, where response time is critical.

The developed system for online handwritten mathematical expressions recognition consists of two stages. In the first stage, the segmentation and classification of symbols are performed using neural networks, and in the second stage, the most probable mathematical expression tree is built.

Symbol segmentation and classification are done with neural network architecture named Bidirectional Long Short–Term Memory. The training of this neural network is built by applying Connectionist Temporal Classification scoring function. Due to the ability to memorize long-term dependencies, this combination is successful in many recognition tasks for one-dimensional languages, where proper segmentation is not required. In this work, these technologies are adapted to solve two problems simultaneously, namely character segmentation and classification. The precise segmentation of symbols is required for the recognition of mathematical expressions, as well as the proper classification because the result of segmentation affects the following stages. An additional lightweight neural network for the classification of individual symbols and a corresponding algorithm for integrating the results of two neural networks were proposed to improve the symbol segmentation and classification quality. This algorithm provides classification results refinement considering the possible character segmentation variants. An additional neural network reduced recognition errors at the expression level by $\approx 10\%$.

The construction of the mathematical expression tree is carried out using a two-dimensional stochastic context-free grammar and parsing algorithm. The parsing

method is based on the Cocke–Younger–Kasami algorithm and uses the adapted version for two-dimensional languages. A set of geometric features was proposed to solve the problem of spatial relations classification between the elements of a mathematical expression. Also, the «body box» concept was improved through the use of local context. Based on the accuracy and speed of different algorithms for spatial relation classification, a decision tree algorithm was selected, as this algorithm is more in line with the requirements for limited computing resources.

The probability of each mathematical expression hypothesis is calculated using an integration function that combines the results from several classifiers: segmentation and classification of symbols, spatial relations, language model, and grammar models. Grammatical parsing methods require high computing resources and can have factorial time complexity. In this work, a set of methods was proposed to reduce the time complexity of the parsing algorithm. These methods are based on computational geometry, graph theory, and dynamic programming. Integration dynamic beam–width adjustment for beam search algorithm allows avoiding the disadvantages of greedy algorithms. Construction of function names at an early stage reduces the length of the input sequence. Geometric search methods minimize complexity by choosing the proper set of elements during the mathematical expression tree construction. The dominance tree reduces the complexity of the algorithm with vertically arranged elements. Coverage regions prevent the construction of redundant intermediate hypotheses.

One of the problems in artificial neural network-based systems development is the need for large data sets. The preparing datasets task for the handwritten mathematical expressions recognition is complicated by the need for detailed annotation of these sets. The annotation of mathematical expressions has a hierarchical structure and consists of several levels. Each annotation level is used to train different models and evaluate the relevant stages of the recognition system. This work describes a new approach to data annotation automation for online handwritten mathematical expression recognition based on recurrent neural networks that do not require initial verification of samples or pre-prepared models. The method demonstrated high annotation accuracy, which was almost 99% at the expression level. More than 85% of the examples have been

automatically annotated. As a result, the amount of manual work has decreased by more than 90%. The proposed approach can also be applied to high-noise datasets and to identify samples that require expert judgment.

CROHME open benchmarks were used to validate the accuracy of the proposed method, and recognition speed was measured on mobile devices. We used the recognition accuracy metric for the entire expression to compare with other systems. This metric is very sensitive to any errors. The proposed method demonstrates high accuracy. The expression rates for the 101-character alphabet are 73.12%, 71.75% and 72.31% at CROHME 2014, 2016, and 2019 respectively. Evaluation of the methods for reducing computational complexity was carried out in a series of experiments using the ablation study. The proposed methods reduced the structural analysis time from 5–7 seconds to 20–40 milliseconds. As a result, the average expression recognition time on a mobile device is about 55 milliseconds.

The system, which is based on the methods proposed in this work, participated in the open competition on recognition of online handwritten mathematical expressions in 2019. The proposed version of the system demonstrated 79.82% expression recognition accuracy and won second place. The system was also modified to participate in the competition on offline recognition and spotting of handwritten mathematical expressions in 2020 and showed 61.90% expression rate.

The proposed system has been successfully integrated into several mobile applications with a pen-based interface. The presented applications are designed to fast and natural way input complex documents. A handwritten calculator for input, editing, and solving mathematical expressions demonstrates the ability to integrate the proposed method for constructing iterative management of mathematical expressions. The example of the application «Interactive Paper» shows an example of a smart pen-based interface for inputting complex documents that contain text, mathematical expressions, tables, graphs, and charts. The usability of the sketch-based interface has been explored by performing tasks with a relatively large number of participants. The method was integrated into the commercial application «S Note» for *Samsung Galaxy Note 9* as a

separate plugin for transforming a set of handwritten strokes into a printed representation of a mathematical expression.

This dissertation is a part of the research conducted at the Department of Mathematical Informatics at the Faculty of Computer Sciences and Cybernetics of Taras Shevchenko National University of Kyiv.

Key words: handwritten mathematical expression, online recognition, recurrent neural network, context-free grammar, human-computer interaction, pen-centric interface, ubiquitous and mobile computing

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

За результатами дослідження опубліковано 8 наукових праць: 1 – у закордонних періодичних наукових фахових виданнях, що індексуються наукометричною базою Scopus; 1 – у вітчизняному фаховому виданні; 6 – у матеріалах доповідей міжнародних конференцій та тез, що індексуються наукометричними базами Scopus та Web of Science.

Статті в іноземних виданнях (тільки закордонні періодичні (ISSN) наукові фахові видання):

[1] Dmytro Zhelezniakov, Viktor Zaytsev, and Olga Radyvonenko. 2021. Online Handwritten Mathematical Expression Recognition and Applications: A Survey. *IEEE Access* 9 (2021), 38352–38373. https://doi.org/10.1109/ACCESS.2021.3063413

Статті в інших виданнях (статті у неперіодичних збірниках наукових праць (ISBN), у т.ч. статті у збірниках за матеріалами конференцій):

[1] Anastasiia Cherneha, Dmytro Zhelezniakov, Pavlo Tytarchuk, and Vasyl Tereshchenko. 2021. Segmentation of Handwritten Mathematical Matrices Using the Area Voronoi Diagram. In *IEEE EUROCON 2021 - 19th International Conference on Smart Technologies* (Lviv, Ukraine). 107–112. https://doi.org/10.1109/EUROCON52738.2021.9535572

[2] Oleg Yakovchuk, Anastasiia Cherneha, Dmytro Zhelezniakov, and Viktor Zaytsev. 2020. Methods for Lines and Matrices Segmentation in RNN-based Online Handwriting Mathematical Expression Recognition Systems. In *2020 IEEE Third International Conference on Data Stream Mining Processing (DSMP)* (Lviv, Ukraine). 255–261. https://doi.org/10.1109/DSMP47368.2020.9204273

[3] Dmytro Zhelezniakov, Anastasiia Cherneha, Viktor Zaytsev, Tetiana Ignatova, Olga Radyvonenko, and Oleg Yakovchuk. 2020. Evaluating New Requirements to Pen-Centric Intelligent User Interface Based on End-to-End Mathematical Expressions Recognition. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (Cagliari, Italy) *(IUI '20)*. Association for Computing Machinery, New York, NY, USA, 212–220. https://doi.org/10.1145/3377325.3377482

[4] Dmytro Zhelezniakov, Anastasiia Cherneha, Viktor Zaytsev, and Olga Radyvonenko. 2021. A New Approach to Data Annotation Automation for Online Handwritten Mathematical Expression Recognition based on Recurrent Neural Networks. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (Melbourne, Australia). 1125–1132. https://doi.org/10.1109/SMC52423.2021.9658867

[5] Dmytro Zhelezniakov, Viktor Zaytsev, and Olga Radyvonenko. 2019. Acceleration of Online Recognition of 2D Sequences Using Deep Bidirectional LSTM and Dynamic Programming. In *Advances in Computational Intelligence*. Springer International Publishing, Cham, 438–449. https://doi.org/10.1007/978-3-030-20518-8_37

Статті у наукових фахових виданнях України (які входять до переліку ВАК/МОН України):

[1] Oleksandr Marchenko, Olga Radyvonenko, Tetiana Ignatova, Pavlo Titarchuk, and Dmytro Zhelezniakov. 2020. Improving text generation through introducing coherence metrics. *Cybernetics and Systems Analysis* 56, 1 (2020), 13–21. https://doi.org/10.1007/s10559-020-00216-x

Тези наукових доповідей:

[1] Dmytro Zhelezniakov, Viktor Zaytsev, Olga Radyvonenko, and Yevhenii Yakishyn. 2019. InteractivePaper: Minimalism in Document Editing UI Through the Handwriting Prism *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 13–15. https://doi.org/10.1145/3332167.3357099

Міжнародні патенти:

[1] Dmytro Zhelezniakov, Oleksandr Marchenko, Ivan Deriuga, Olga Radyvonenko, Pavlo Tytarchuk, Volkova Volkova, Viktor Zaytsev, and Tetiana Ignatova. 2021. Electronic device and story generation method thereof. https://patents.google.com/patent/US20210224310A1/en US Patent 224,310.

[2] Dmytro Zhelezniakov, Viktor Zaytsev, Olga Radyvonenko, and Pavlo Tytarchuk. 2021. Electronic device for converting handwriting input to text and method of operating the same. https://patents.google.com/patent/US20210150200A1/en US Patent 150,200.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1D	Одновимірний (One–Dimensional)
2D	Двовимірний (Two–Dimensional)
2D-СКВГ	Двовимірна стохастичних контекстно-вільна граматика (Stochastic Context-Free Grammars, 2D-SCFG)
ВРВ	Вентильний рекурентний вузол (Gated recurrent units, GRU)
ГН	Глибинне навчання (Deep Learning, DL)
ДДКЧП	Двонаправлена довга короткочасна пам’ять (Bidirectional Long Short-Term Memory, BLSTM)
ДКЧП	Довга короткочасна пам’ять (Long Short-Term Memory, LSTM)
ДРНМ	Двонаправлена рекурентна нейронна мережа (Bidirectional Recurrent Neural Networks, BRNN)
ЗНМ	Згорткові нейронні мережі (Convolutional Neural Networks, CNN)
КВГ	Контекстно-вільна граматика (Context-Free grammar, CFG)
КЯК	Кокке-Янгер-Касамі (Cocke-Younger-Kasami, CYK)
МВ	Математичний вираз (Mathematical Expression, ME)
МКД	Мінімальне кістякове дерево (Minimum Spanning Tree, MST)
ММ	Модель мови (Language Model, LM)
МН	Машинне навчання (Machine Learning, ML)
НМ	Нейронна мережа (Neural Network, NN)
НФЧ	Нормальна форма Чомскі(Chomsky Normal Form, CNF)
НЧК	Нейромережева часова класифікація (Connectionist Temporal Classification, CTC)
РМВ	Рукописний математичний вираз (Handwrittent Mathematical Expression, HME)

РНМ	Рекурентна нейронна мережа (Recurrent Neural Networks, RNN)
СКВГ	Стохастичних контекстно-вільна граматика (Stochastic Context-Free Grammars, SCFG)
ТРВ	Точність розпізнавання виразів (Expression Rate, ER)
ТРДВ	Точність розпізнавання дерева виразу (Structure Rate, SR)

ВСТУП

Актуальність теми. У сучасному світі ввід інформації найчастіше асоціюється з клавіатурним або оптичним вводом, де зображення для обробки можуть бути отримані за допомогою цифрових камер, сканерів, мікроскопів тощо. Але, починаючи з 2007 року, вибухове зростання користувачів смартфонів (зараз оцінюється в понад 6 мільярдів) призвело до масштабного створення контенту користувачів, при цьому змінився основний спосіб вводу інформації. Так, для вводу звичайного тексту, який є прикладом одновимірної (1D) мови, прийшли на допомогу віртуальна клавіатура і технології в галузі розпізнавання голосу. Але ці технології не забезпечують зручного вводу більш складних документів, які містять візуальні або двовимірні мови (2D). Діаграми, хімічні рівняння, музичні ноти, математичні вирази та електричні схеми – це приклади так званих двовимірних мов. Так, для вводу двовимірних елементів документів було запропоновано стилус – цифровий аналог ручки. Для користувача введення інформації за допомогою такого інтерфейсу є більш природним та продуктивним. Таким чином, виникає проблема розпізнавання рукописного вводу. При цьому слід враховувати значні обмеження обчислювальної потужності сучасних мобільних пристроїв порівняно із звичайним комп’ютером та вимоги щодо низького енергоспоживання. Тому розробка високоефективних методів розпізнавання двовимірних мов є надзвичайно актуальним напрямком досліджень. Такі методи дають можливість створювати нові інтелектуальні інтерфейси для взаємодії людини з комп’ютером на основі пера або стилуса.

Ця робота зосереджена на онлайн розпізнаванні рукописних двовимірних мов, а саме математичних виразів (МВ), оскільки математичний вираз є найцікавішим з точки зору використання та являє собою одну з найскладніших мов з погляду розпізнавання, оскільки містить понад 1500 символів [19]. Найчастіше ввід МВ

здійснюється за допомогою систем комп'ютерної верстки, таких як LaTeX, або редакторів математичних виразів, таких як MathType. Такі системи вимагають від користувача запам'ятовувати велику кількість команд для вводу символів та синтаксичних правил або постійної навігації серед десятків елементів меню для пошуку необхідного символу або конструкції виразу.

Розпізнавання МВ відносять до задач розпізнавання образів (*pattern recognition*). Цю задачу можна розглядати з різних точок зору: "Друковані або рукописні"та "Онлайн або офлайн". Онлайн розпізнавання працює з динамічним представленням вхідних даних (сліди руху пера/пальця або векторне зображення), а офлайн розпізнавання отримує на вхід статичне растрове зображення. Розпізнавання 2D мов є більш складним завданням порівняно з розпізнаванням одновимірних мов.

Перші роботи в галузі розпізнавання математичних виразів з'явилися в другій половині 1960-х років [10, 28]. Роботи 1960–80-х років в основному були зосереджені на розпізнаванні друкованих виразів у зображенні, вони заклали основу для подальшого розвитку. З еволюцією та широким розповсюдженням пристроїв із сенсорним екраном та електронним пером на початку 2000-х років, таких як Pocket PC, інтерес до рукописного вводу та онлайн розпізнавання значно зріс. Сучасні підходи засновані на «sequence-to-sequence» глибинному машинному навчанні (*Deep Machine Learning, DML*) та призвели до епохального підвищення точності розпізнавання. В останні роки розробляються і переважають «end-to-end» підходи, основним обмеженням яких є потреба в потужних обчислювальних ресурсах [191, 192, 193], тому вони не завжди можуть бути застосовані на мобільних пристроях.

Зв'язок роботи з науковими програмами, планами, темами. Робота є складовою частиною наукових досліджень, які ведуться на кафедрі математичної інформатики факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка на виконання теми проекту Міністерства освіти і науки України «Розробка єдиного програмно-алгоритмічного се-

редовища візуалізації та комп'ютерного моделювання для створення систем оздоровлення військовослужбовців» (НДФ №19БФ015-04, 01.01.2019-31.12.2021)

Мета і завдання дослідження. Онлайн розпізнавання 2D мов може розглядатися як перетворення рукописних штрихів у відповідну нотацію. Так, для математичних виразів прикладом таких нотацій є LaTeX або MathML.

Метою дисертаційного дослідження є створення моделі ефективного розв'язування комплексу задач розпізнавання рукописних математичних виразів. Для досягнення поставленої мети необхідно розв'язати такі завдання:

1. Розробити загальну архітектуру програмного рішення для розв'язування задач розпізнавання РМВ.
2. Створити на основі розробленої архітектури засоби для розв'язування таких задач: попередня обробка та кластеризація вхідних даних, класифікація послідовності рукописних об'єктів, класифікації просторових відношень, побудова дерева відношень між елементами виразу, побудова результату за допомогою загальноприйнятих форматів таких як LaTeX.
3. Розробити ефективні алгоритми розв'язування задач розпізнавання РМВ та покращити існуючи на сьогодні методи.
4. Дослідити методи класифікації послідовності рукописних об'єктів на основі рекурентних нейронних мереж.
5. Дослідити методи побудови ймовірнісної контекстно-вільної граматики для двовимірних мов та класифікації просторових відношень.
6. Розробити ефективні методи побудови дерева відношень між об'єктами з використанням ймовірнісної контекстно-вільної граматики.
7. Дослідити та покращити методи побудови анотованих наборів даних для задач розпізнавання рукописного вводу.
8. Виконати програмну реалізацію моделі розпізнавання математичних виразів, що включає оптимізацію роботи алгоритмів та тестування ефективності роботи алгоритмів.
9. Провести обчислювальні експерименти на відкритих наборах даних CROHME.

10. Продемонструвати практичну цінність розробленої моделі на прикладі мобільних додатків, що забезпечують вирішення різних задач.

Об'єкт дослідження. Процес ефективного розв'язання задач розпізнавання рукописних математичних виразів та побудова інтерфейсу користувача з пером для вводу складних документів.

Предмет дослідження. Методи, алгоритми та структури даних для ефективного розв'язування задач розпізнавання двовимірних мов.

Методи дослідження. Дослідження ґрунтуються на методах машинного навчання, обчислювальної геометрії, формальних граматик, теорії графів та динамічного програмування.

Наукова новизна отриманих результатів. Основним напрямком цього дослідження є створення нового методу розпізнавання РМВ, який може застосовуватися на мобільних пристроях та забезпечувати ітеративний режим вводу МВ. У рамках цього дослідження було:

- вперше запропоновано метод одночасної сегментації та класифікації символів для двовимірних мов в рукописній послідовності об'єктів на основі двонаправленої довгої короткочасної пам'яті та нейромережевої часової класифікації, що забезпечує високу точність розпізнавання;
- вперше запропоновано та розроблено метод для підвищення точності маркування послідовностей за допомогою додаткової легковагової нейронної мережі, яка виконує тільки завдання класифікації символів;
- запропоновано новий набір геометричних ознак, який використовує удосконалену концепцію «body box» і обмежувальний прямокутник одночасно, для класифікації просторових відношень між елементами математичного виразу, що дозволило підвищити точність класифікації просторових відношень;
- вперше адаптована та застосована концепція домінуючих символів для граматичних підходів структурного аналізу за рахунок використання дерева домінування та розширення алгоритму "Кокке-Янгер-Касамі", що до-

зволило суттєво зменшити складності алгоритму розбору з вертикально розташованими елементами;

– запропоновано набір підходів для зменшення складності алгоритму структурного аналізу для рукописних виразів за рахунок використання обчислювальної геометрії, евристичних методів та методів динамічного програмування;
– вперше запропоновано ітеративний метод для напівавтоматичної анотації рукописних математичних виразів, що дозволяє значно зменшити час підготовки наборів даних для тренування та тестування.

Практичне значення отриманих результатів. Розроблена система онлайн розпізнавання РМВ дає змогу розв'язувати такі задачі: сегментація та класифікація символів, класифікації просторових відношень, побудова 2D структури математичного виразу. Також запропоновані методи можуть застосовуватися для онлайн розпізнавання скетчів, музичних нот, діаграм та інших 2D мов. Запропонований метод для сегментації та класифікації символів забезпечує високу точність сегментації, що дає можливість використовувати його для інтерактивного редагування МВ. Більше того, представлений метод сегментації та класифікації символів може застосовуватися в задачах офлайн розпізнавання за умови перетворення вхідного зображення на набір штрихів. Приклад такого рішення та результати наведено у Додатку А. Методи класифікації просторових відношень та структурного аналізу є універсальними та можуть використовуватися у вирішенні задач розпізнавання як рукописних, так і друкованих МВ.

Розроблений метод демонструє високі показники якості та високу швидкодію, що дає змогу застосовувати його на пристроях з обмеженнями. У дисертаційній роботі було продемонстровано застосування розробленої системи розпізнавання РМВ у таких мобільних додатках:

– прототип рукописного калькулятора для вводу, редагування та обчислення математичних виразів;
– «Інтерактивний папір» для вводу за допомогою пера документів зі складною структурою;

– «S Note» для створювання заміток.

Тенденції та перспективні задачі. Розроблені підходи до глибинного навчання призвели до кардинального підвищення точності в розпізнаванні послідовностей та задачах моделювання. Водночас набули значного поширення пристрої для введення інформації за допомогою пера, такі як мобільні телефони, інтерактивні дошки тощо. Цей прогрес разом із зростанням потреб користувачів відкриває нові можливості для подальших досліджень.

Додатки з інтерфейсом "Папір і олівець" (pen-based або sketch-based applications). Найбільш широко представлені навчальні додатки [114, 117, 144, 157] з розпізнаванням РМВ, оскільки математика найбільш потрібна в них. Однак сучасний стан технології дає можливість застосовувати такий інтерфейс у зростаючий кількості галузей, таких як хімія, фінанси та інші.

Рішення на пристрої (on-deivce solutions). Підходи до мобільних платформ зараз користуються попитом. Виробники мобільних пристроїв часто фокусуються на розробці рішень, що не потребують постійного доступу до хмарних сервісів. Такі рішення є більш мобільними та викликають довіру користувачів.

Персоналізація. Ця сфера розпізнавання МВ ще не розроблена. Такі прийоми, як навчання з підкріпленням та few-short навчання, можуть зробити систему більш гнучкою та забезпечити можливість адаптації до конкретного користувача. Це особливо важливо для систем розпізнавання мов з великим розміром алфавіту. Розробка підходів, що до персоналізації допоможе значно зменшити неоднозначність, пов'язану з особливостями рукописного вводу кожного користувача.

Контекстно-залежні підходи. Контекстно-залежні методи розпізнавання МВ також не вивчені. Переміщення фокусу з розпізнавання єдиного МВ на підтримку вводу декількох МВ з урахуванням контексту документа допоможе уникнути багатьох неоднозначностей, пов'язаних із подібністю багатьох математичних символів та складністю 2D-структури.

Ітераційний ввід даних. Незважаючи на значне збільшення якості розпізнання, додатки, які підтримують рукописний ввід, мають бути побудовані з урахува-

нням ітераційного вводу даних та з можливістю внесення змін до вже існуючих елементів.

Наскрізні рішення. В той час як набір методів для послідовних та інтегрованих рішень досить глибоко вивчався в останні десятиліття, наскрізні (end-to-end) рішення для розпізнавання математичних формул почали розвиватися лише декілька років тому. Але нинішня складність наскрізних рішень є занадто обтяжливою для багатьох пристроїв і вимагає хмарних обчислень. Крім того, наскрізні рішення фокусуються на розпізнаванні цілого виразу, тож адаптація таких підходів до ітеративного вводу розширить сферу застосування цих рішень.

Хмарні рішення. Останнє CROHME змагання показало, що поєднання методів онлайн та офлайн розпізнавання забезпечує значне підвищення точності розпізнавання. Крім того, очікується, що хмарні рішення зможуть значно покращити якість розпізнавання за рахунок федеративного навчання. Але хмарні рішення пов’язані з ризиками в галузі захисту приватних даних.

Мультимодальність. Мультимодальність активно розвивається в багатьох сферах, таких як Visual Question Answering. Крім того, користувачі мають різні переваги щодо вводу інформації залежно від поточної активності [184]. Таким чином програми дають можливість безперешкодно перемикатися між клавіатурою/мишкою та рукописним текстом, забезпечуючи підтримку кількох модальностей. Голосове управління також може брати активну участь у розробці інтерфейсу користувача.

Інтегровані рішення. Ці методи найбільш вивчені, але з року в рік все ще демонструють значний прогрес.

Складність МВ. У цей час комерційні системи підтримують розпізнавання менш ніж 200 різних типів символів. Перевірка підходів проводиться на відкритих наборах даних, що містять лише 101 тип символів, тоді як математична нотація має більш ніж 1500 символів. Існуючі рішення забезпечують загальну модель розпізнавання, яка не залежить від конкретної галузі, хоча багато наукових та інженерних дисциплін адаптували математичні позначення відповідно до своїх потреб. Правила побудови виразів, підмножина символів у багатьох галузях сут-

тєво відрізняються. Таким чином, очікується, що кількість символів, типів виразів і кількість областей буде збільшуватися.

Виявлення МВ. Одним з основних завдань для дослідників є виявлення МВ для забезпечення “безшовного” вводу та розпізнавання різних елементів документа (тексту, математики, таблиць, ескізів та інших) під час рукописного вводу. Поточні дослідження в основному зосереджені на локалізації математичних виразів у друкованому тексті, що не передбачає ітеративного вводу. Така ситуація призводить до створення складних інтерфейсів і не дає можливість повною мірою запроваджувати концепцію “Папір і олівець”.

Змішана та доповнена реальність. Існує потенціал у взаємодії з математичними виразами в доповненій та змішаній реальності. Зокрема, це може призвести до відкриття багатьох можливостей для підвищення продуктивності освіти. Поєднання екранного вводу ручкою чи дотиком та розпізнавання РМВ з редагуванням на основі погляду та дотику, доповнене візуалізацією залежностей або результатами прогнозування/авто-доповнення, може покращити взаємодію з користувачем та виявити нові проблеми в розпізнаванні МВ.

РОЗДІЛ 1

ПОСТАНОВКА ЗАДАЧІ ТА ЗАГАЛЬНА АРХІТЕКТУРА ЗАПРОПОНОВАНОГО РІШЕННЯ

Математичні вирази є важливою складовою в багатьох галузях таких як наука, фінанси, освіта та інші. МВ відрізняються від текстового представлення наявністю двовимірної (2D) структури та великого розміру алфавіту (понад 1500 символів [19]), де символи дуже часто схожі між собою, особливо для рукописних МВ.

Користувачі здебільшого віддають перевагу рукописному введенню МЕ, ніж клавіатурі та миші, що набагато повільніше [11]. Незважаючи на багатообіцяльні нові розробки в розпізнаванні РМВ, це завдання все ще знаходиться на тому рівні, коли помилки розпізнавання трапляються досить часто. Хороший інтерфейс користувача значно покращує систему, тоді як неефективний може зіпсувати враження від системи навіть із майже ідеальною точністю розпізнавання РМВ.

1.1. Проблеми розпізнавання рукописних математичних виразів

Найчастіше розпізнавання РМВ порівнюють із розпізнаванням рукописного тексту. Розпізнавання РМВ є набагато складнішим завданням, головним чином через 2D структуру.

Загалом, проблема онлайн-розпізнавання МВ може бути сформульована як перетворення рукописного вводу у деревоподібне представлення, таке як MathML або LaTeX. МВ представляється як набір штрихів, що впорядковані в порядку написання. Кожен штрих – це послідовність точок, які повторюють траєкторію написання. Кожен штрих починається в момент дотику (*pen-down*) та закінчується в момент піднімання пера (*pen-up*). Рисунок 1.1 демонструє приклад математичного виразу, що містить 18 штрихів.

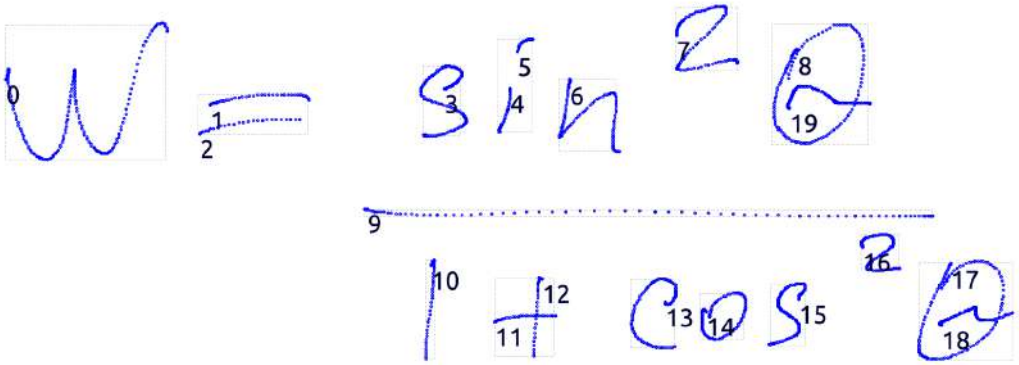


Рис. 1.1. Приклад рукописного математичного виразу $w = \frac{\sin^2\theta}{1+\cos^2\theta}$

Розпізнавання МВ традиційно складається з двох етапів [25]: 1) розпізнавання символів та 2) структурний аналіз. Етап розпізнавання символів містить такі завдання: 1) попередня обробка штрихів, яка включає різні техніки, такі як нормалізація розміру, інтерполяція штрихів, корекція нахилу та передискретизація; 2) сегментація символів або групування вхідних штрихів, які належать одному символу; 3) класифікація символів або маркування груп штрихів. Метою структурного аналізу є виявлення просторових зв'язків між елементами, пошук математичної інтерпретації виразу та отримання його у вигляді математичної нотації. Рисунок 1.2 ілюструє класичний процес розпізнавання МВ. Автори статті [185] розглянули питання розпізнавання МВ в контексті розпізнавання документів та виділили додаткову задачу: виявлення МВ у документі.

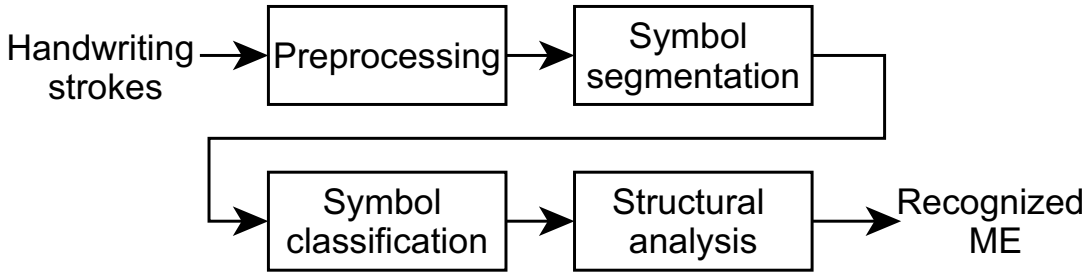


Рис. 1.2. Процес розпізнавання в послідовних рішеннях

З кожним етапом пов'язані свої проблеми, які позначаються на якості та швидкості розпізнавання та роблять цю задачу дужу складною. Різні пристрої надають різну швидкість зчитування точок під час написання. Також кількість точок силь-

но залежить від швидкості написання, що добре видно на Рисунку 1.1. А розмір символів залежить від роздільної здатності екрана та особливостей написання кожного користувача. Також у траєкторії написання можуть міститися шуми, що викликані тремтінням пера. На відміну від рукописного тексту, у МВ розміри символів можуть значно відрізнятися залежно від положення у виразі. Зазвичай розміри символів зменшуються зі збільшенням рівня підвиразу. Завданням **попередньої обробки** є нормалізація вхідних даних з точки зору розмірів символів, частоти точок та мінімізації шумів. При цьому важливо не втратити значущої інформації.

Завданням **сегментації символів** є визначення груп штрихів, що утворюють один символ. У друкованому вигляді математичні символи мають чіткі інтервали між символами. У разі рукописних МВ символи часто перетинаються або, навпаки, штрихи, що належать одному символу, можуть мати значну відстань. Задача ускладнюється тим, що в написанні порядок штрихів не визначений і користувач може дописувати частини символу у будь-який момент часу. Рисунок 1.1 містить приклад написання символу θ, що складається з двох штрихів із номерами 8 та 19.

Класифікація символів у математичному виразі – дуже складна задача як для друкованого представлення, так і для рукописного. Головною проблемою є великий розмір алфавіту, через що деякі вирази надзвичайно важко розпізнати навіть людині. Найпоширенішими прикладами є двозначності, що пов'язані з дуже схожим або навіть однаковим написанням багатьох символів. Існує багато символів, що мають однакове написання у нижньому та верхньому регістрі. Прикладами таких символів можуть бути "X/x", "C/c", "K/k" та багато інших. Наявність латинських, грецьких та римських символів також додає плутанини через схожість багатьох із них, таких як "B/β", "p/ρ", "x/χ", and "n/η". Неоднозначність позначень також існує між символами та операторами, такими як "$x/\times$", "$1/|$", "$S/\int$", "$o/\circ$", "$t/+$", "$\angle/<$". Написання кількох символів поруч може також призвести до неоднозначності в сегментації та класифікації символів. Ось кілька прикладів таких двозначностей: "$)($" – "x", "13" – "B", "$1<$" – "K". Неоднозначність виникає не лише через велику кількість дуже схожих символів, а й через особливості

почерку кожної людини чи регіональну специфіку написання. До регіональних відмінностей належать також відмінності у математичних позначеннях. Для прикладу, деякі країни мають свої власні правила найменування функцій.

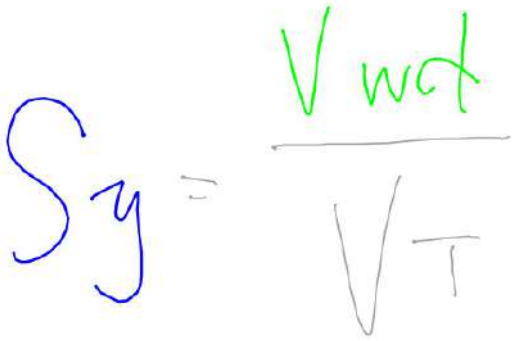
Крім неоднозначностей, пов'язаних із сегментацією та класифікацією символів, наступною серйозною проблемою є класифікація просторових відношень та **структурний аналіз.** У математичній нотації просторові відношення здебільшого позначають неявні оператори. Хоча кількість просторових зв'язків невелика, відношення між елементами можуть бути дуже нечіткими. В математичних записах часто використовується оператор неявного множення, що також може створити певні труднощі у визначенні правильної структури виразу. Приклади різних неоднозначностей, що пов'язані із сегментацією, класифікацією та аналізом структури, показані на Рисунку 1.3.

Підготовка наборів даних для навчання та верифікації також є складним завданням, оскільки вимагає збору, перевірки та маркування чималого набору даних. Багато підходів потребують значної ручної праці для анотування зібраних наборів даних по символах. Також система розпізнавання має будуватися не тільки з точки зору якості розпізнавання, але й зручності інтерфейсу користувача.

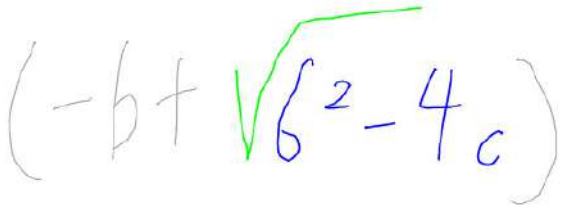
Застосування на мобільних пристроях створює свої виклики у розробці системи. З одного боку продуктивність мобільних пристроїв наближається до продуктивності персональних настільних систем нижнього цінового діапазону. Але при цьому вже досить тривалий час не змінюється місткість батареї. Багато виробників мобільних пристроїв ставлять час автономної роботи як одне з головних завдань. Насамперед низьке споживання енергії досягається за рахунок впровадження ефективних алгоритмів. Другою проблемою є споживання постійної пам'яті. Це викликано тим, що виробники вважають за краще відразу встановити більшу кількість функцій для залучення клієнтів. Тобто в сучасних умовах, купуючи мобільний пристрій, користувач насамперед отримує не процесор з пам'яттю, а набір сервісів, які надає цей пристрій. Рахунок на кількість нейронних моделей, які користувач отримує одразу під час покупки пристрою може рахуватися вже сотнями. Прикладами таких моделей можуть бути розпізнавання голо-

(а) Неоднозначне написання цифри “9”.

(б) Приклад написання цифри “7” в окремих регіонах.



(в) Невизначеність у визначенні просторових відношень: Sy або S_y, Vwd або V_{wd}



(г) Неоднозначність в виразі з радикалом: $\sqrt{b^2} - 4c$ або $\sqrt{b^2 - 4c}$

(д) Невизначеність сегментації : $1 < b$ або kb

(е) Приклад явного та неявного множення в одному виразі: $\frac{4}{3}x - \frac{2}{3} \times (465 - x)$

Рис. 1.3. Приклади невизначеностей у розпізнаванні РМВ

су, визначення обличчя під час зйомки, пошук об’єктів на фотографії та багато інших. Все це веде до того, що розмір пам’яті, доступний для зберігання контенту кінцевого користувача, зменшується. А збільшення розміру пам’яті пристрою призводить до значного подорожчання. На допомогу часто приходять хмарні обчислення. Але вони викликають занепокоєння щодо конфіденційності [32] і тому не завжди можуть використовуватися для роботи з контентом користувача, особливо документами.

1.2. Основні означення

У цій секції наведено основні поняття, що використовуються для розробки системи розпізнавання РМВ.

Означення 1.1. *Математичний вираз* – кінцева комбінація символів, що має двовимірну структуру та правильно побудована згідно з правилами, які залежать від прикладної галузі. Математичні символи можуть означати числа (константи), змінні, операції, функції, пунктуацію, групування та інші аспекти логічного синтаксису.

Означення 1.2. *Структура математичного виразу* або *дерево розбору* представляється графічною моделлю, наприклад, орієнтованим або неорієнтованим графом. Вузли графіка відповідають окремим символам, а ребра – просторовим відношенням між сусідніми парами символів або підвиразами. Більш детальна інформація про дерево розбору та алгоритм його побудови подана в Розділі 5.

Означення 1.3. *Підвираз* є частиною виразу, який відповідає піддереву в дереві структури математичних виразів, тобто деякому вузлу в дереві з усіма його нащадками. Здебільшого підвираз є правильним математичним виразом, якщо це не весь вираз.

Означення 1.4. *Просторове відношення* вказує, як деякий об'єкт розташований у просторі відносно до якогось опорного об'єкта [50]. Просторові відношення у математичній нотації визначаються позицією та відносним розміром символів у виразі.

Означення 1.5. Вхідні дані для онлайн розпізнавання РМВ позначаються як:

$$S = \{S^m, m = 1, \ldots, M\}$$

де: S^m представляє штрих з номером m, а M – це кількість вхідних штрихів.

Означення 1.6. Штрих подається у вигляді часового ряду точок:

$$S^m = \{p_n^m, n = 1, \ldots, N^m\}$$

де: $p_n^m = (x_n^m, y_n^m)$ – це координати пера точки n, а N^m – це загальна кількість точок у штриху.

Означення 1.7. LaTeX – мова розмітки даних та пакет макросів TeX для високоякісного оформлення документів, який представляє стандартизований формат для запису математичних виразів за допомогою послідовності команд. Приклад математичного виразу у форматі LaTeX: *e^{n+1}* (e^{n+1}).

Означення 1.8. *MathML* – мова розмітки для опису математичних виразів, що основана на XML. Застосовується для розміщення математичних формул на сторінках інтернету.

Означення 1.9. *Базова лінія (baseline)* – уявна лінія, на якій розташовуються літери одного рівня і яка проходить по основі символів. Для прикладу: МВ $a + \frac{b}{c}$ має 3 базові лінії.

Означення 1.10. Під *контекстом розпізнавання* ми маємо на увазі суміжні символи та їхні семантичні ознаки.

Означення 1.11. Мінімальним *результатом роботи* системи розпізнавання є дерево розбору або МВ, представлений в одній із мов розмітки. Результат роботи може містити варіанти розпізнавання разом з їх оцінками, а також додаткову інформацію про геометричні ознаки кожного символу та підвиразу, включаючи результат сегментації, класифікації просторових відношень тощо.

1.3. Опис основних задач та вимог до системи розпізнавання РМВ

У роботі особлива увага приділяється таким задачам розпізнавання РМВ:

Задача 1. *Сегментація символів* полягає у визначенні штрихів та/або діапазону точок, що утворюють один символ.

Задача 2. *Класифікація символів* полягає у прогнозуванні класів (кодів) математичних символів для заданої послідовності вхідних штрихів або точок.

Задача 3. *Класифікація просторових відношень* полягає у визначенні просторових відношень та їх ймовірності між символами або підвиразами в рукописному математичному виразі.

Задача 4. *Структурний аналіз* полягає у побудові дерева розбору, що містить інформацію про символи, їхні геометричні ознаки та просторові відношення між підвиразами. Зокрема, таке дерево використовуватиметься для знаходження найбільш ймовірного представлення математичного виразу.

Задача 5. *Підготовка даних для тренування моделей* полягає в створенні анотованого набору прикладів для тренування моделей для сегментації та класифікації символів, а також для підготовки моделей класифікації просторових відношень. Одним із завдань є створення методів для автоматичної анотації наборів даних на основі наявних наборів даних, але алфавіти цих двох наборів даних відрізняються.

Вимоги до системи розпізнавання РМВ сформульовані з урахуванням застосування системи на мобільних пристроях та забезпечення різних методів побудови інтерфейсів користувача. Далі перелічені основні вимоги:

Вимога 1. *Точність розпізнавання* всього виразу у пакетному режимі має перевищувати параметри наявних систем, тобто складати не менш ніж 67%. Розрахунок точності розпізнавання має здійснюватися на відкритих наборах даних.

Вимога 2. Середній *час розпізнавання* не може перевищувати 1 секунди на сучасному мобільному пристрої при довжині виразу до 10 символів.

Вимога 3. *Розміри моделей та скомпільованої бібліотеки* у сумі не мають перевищувати 5МБ.

Вимога 4. *Пакетний режим* розпізнавання РМВ, при якому результат розпізнавання містить дерево розбору або LaTeX з найбільш ймовірним виразом. У цьому режимі подальше редагування не планується або буде здійснюватися за допомогою клавіатури та миші.

Вимога 5. *Ітеративний режим* розпізнавання РМВ, при якому результат розпізнавання може редагуватися за допомогою вводу рукописних жестів редагування та нових рукописних символів. При цьому побудова виразу здійснюється з набору друкованих символів, які були розпізнані на попередньому етапі, та нових рукописних символів.

1.4. Запропоноване рішення

Підтримка двох режимів розпізнавання істотно ускладнює задачу. Для забезпечення ітеративного режиму необхідно, щоб рішення видавало не тільки результат розпізнавання усього виразу, але і точні геометричні властивості кожного елемента. Під геометричними властивостями мається на увазі інформація про геометричне положення або належність кожного штриха до певного символу. Головною відмінністю цих двох режимів є те, що в пакетному режимі послідовність символів завжди має утворювати кінцевий граматично правильний вираз. А в режимі редагування на вхід може бути така послідовність символів, яка ніколи не зустрічається під час розпізнавання в пакетному режимі і тим паче не може використовуватися для побудови кінцевого правильного виразу або підвиразу. Більшість наявних рішень, зокрема наскрізні рішення на основі згорткових мереж, зосереджені тільки на пакетному режимі.

Двонаправлена довга короткочасна пам’ять (ДДКЧП) разом із нейромережевою часовою класифікацією (НЧК) продемонстрували високі показники якості в багатьох задачах маркування послідовностей, таких як розпізнавання рукописного тексту та мовлення. Багато в чому завдяки можливості ДДКЧП запам’ятовувати довгострокові залежності у вхідній послідовності. Проте при наявності ітеративного режиму редагування неможливо повністю покладатися на можливості нейронної мережі зберігати контекстну інформацію.

Головна ідея цієї роботи полягає в поєднанні можливостей рекурентних нейронних мереж та граматичних методів розбору для забезпечення розпізнавання РМВ. Загальна архітектура запропонованої системи розпізнавання РМВ зображено на Рисунку 1.4. Запропоноване рішення умовно можна поділити на 2 головних етапи. Перший етап (*Character segmentation and classification*) полягає в сегментації та класифікації символів у вхідній послідовності штрихів за допомогою ДДКЧП та НЧК. Одним із головних моментів є правильна попередня обробка вхідних штрихів. Для підвищення якості сегментації та класифікації пропонується інтегрувати додаткову легковагову мережу. Більш детальна інформація про сегмен-

тацію та класифікацію символів представлена у Секції 3. Другий етап (*Structure Analyzer*) забезпечує структурний аналіз та ґрунтується на формалізмі СКВГ із застосуванням класифікатора просторових відношень. Додатково цей етап інтегрує модель мови для підвищення точності розпізнавання. Зазвичай методи КВГ мають досить велику складність, тому для зменшення алгоритмічної складності пропонується набір методів, що спираються на динамічне програмування, методи обчислювальної геометрії та теорії графів. Секції 4 та 5 присвячені класифікації просторових відношень та структурному аналізу.

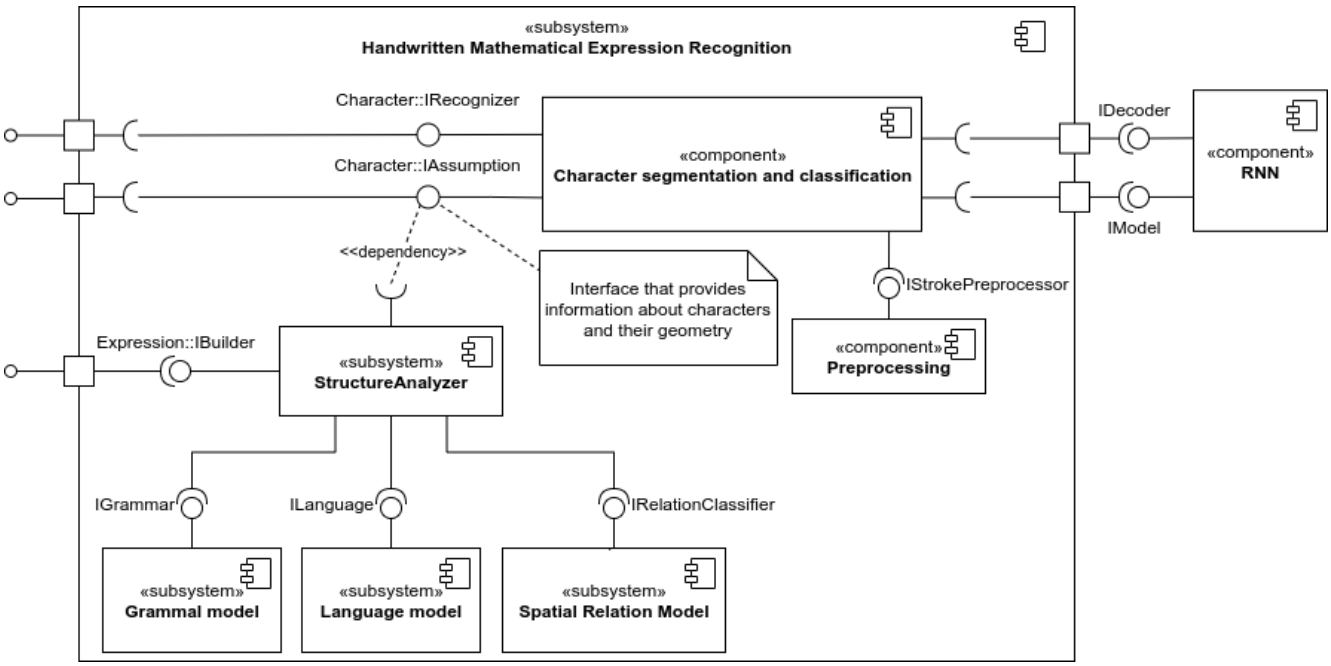


Рис. 1.4. Загальна архітектура програмної реалізації системи розпізнавання РМВ у вигляді діаграми компонент

Однією з головних проблем розробки такого рішення є підготовка наборів даних для тренування та тестування. Наявні відкриті набори даних не задовольняють вимог до розміру наборів даних, що необхідні для тренування нейронних мереж. У Секції 6 розглядається метод автоматичної анотації РМВ з використанням цієї ж системи. Опис експериментів, основні метрики якості та швидкості, порівняння з іншими розробками наведено у Секції 7. Приклади використання запропонованої системи та підходи до побудови інтерфейсу користувача у пакетному та ітеративному режимах представлені у Секції 8.

1.5. Висновки до Розділу 1

Цей розділ присвячений проблемам розпізнавання РМВ. Були розглянуті основні етапи розпізнавання РМВ, що включають попередню обробку, сегментацію та класифікацію символів і структурний аналіз. Проаналізовані та наведені для кожного етапу основні складнощі, що виникають під час процесу обробки. Окремо розглянуті проблеми застосування на мобільних пристроях. Також у цьому розділі подані основні означення та поняття, що використовуються у розробці систем розпізнавання РМВ.

У результаті розгляду основних проблем, пов’язаних із розробкою та застосуванням системи на пристроях з обмеженнями обчислювальних ресурсів та пам’яті, було сформульовано основні задачі та вимоги до системи. Усі завдання безпосередньо пов’язані з процесом розпізнавання та підготовки наборів даних для навчання моделей на основі алгоритмів машинного навчання (МН) та глибокого навчання (ГН). Вимоги включають як якісні показники, так і показники швидкодії та вимоги до постійної пам’яті. Окремо визначені функціональні вимоги щодо режимів системи.

Результатом цього розділу є загальна архітектура системи. Сегментації та класифікації символів побудовані на допомогою ДДКЧП разом із НЧК. Структурний аналіз будується на основі граматичних підходів. Таке об’єднання методів є унікальним та орієнтоване на забезпечення двох режимів розпізнавання (пакетний та ітеративний) з урахуванням обмежень мобільних пристроїв.

РОЗДІЛ 2

АНАЛІЗ НАЯВНИХ ПІДХОДІВ

Дослідження в галузі розпізнавання математичних виразів триває понад 60 років та налічує кілька сотень праць. Поглиблений інтерес до цієї сфери виник у результаті поширення пристроїв із сенсорними екранами та досягнень у галузі машинного навчання.

У цьому розділі розглядається еволюція методів розв'язання завдань в галузі розпізнавання математичних виразів. Найбільш важливими підзадачами є сегментація і класифікація символів та структурний аналіз. Окрему увагу буде приділено наскрізним рішенням та їх порівнянню. Розділ не подає детального опису усіх методів, а тільки робить огляд попередніх робіт. Більш детальний опис наведено у роботі [200].

2.1. Епохи розвитку систем розпізнавання математичних виразів

Жанг вказала три епохи розвитку систем розпізнавання МВ [195]: послідовні рішення, інтегровані рішення та наскрізні рішення на основі НМ (Рисунок 2.1). Послідовні рішення характеризуються тим, що результат попереднього етапу застосовується на наступному. Це призводить до поширення та накопичення помилок. В інтегрованих рішеннях генерується набір гіпотез символів, а модуль структурного аналізу використовує найкращого кандидата символів для побудови належного МВ, враховуючи граматичні та семантичні знання. Наскрізні рішення перетворюють зображення або набір штрихів на вході безпосередньо в математичну нотацію. Тоді як набір методів і прийомів послідовних та інтегрованих рішень вивчався досить глибоко протягом останніх десятиліть, наскрізні рішення для розпізнавання МВ почали розвиватися зовсім недавно. Наскрізні рішення мають велику обчислювальну складність та не надають інформацію про точне роз-

ташування символів. Тому такі рішення не можуть бути застосовані в завданнях редагування, коли необхідно вносити зміни до вже розпізнаних виразів.

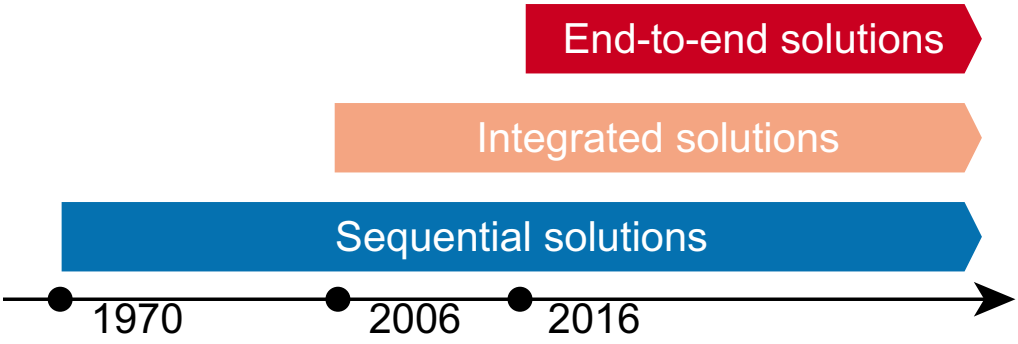
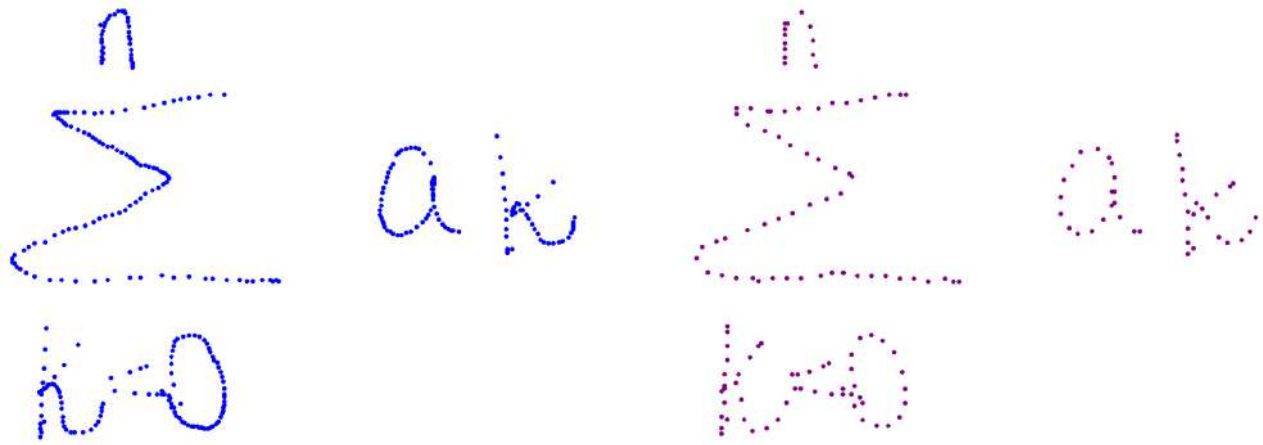


Рис. 2.1. Епохи еволюції систем розпізнавання математичних виразів

2.2. Методи сегментації та класифікації символів

Попередня обробка штрихів. Попередня обробка є першим кроком у задачі розпізнавання РМВ і суттєво впливає на точність моделі розпізнавання. Головною метою цього кроку є уніфікація вхідних даних з метою підвищення надійності алгоритму розпізнавання для вхідних даних. Як правило, попередня обробка рукописних штрихів в онлайн розпізнаванні включає: видалення дубльованих точок та гачків, корекцію нахилу, згладжування вхідних точок, передискретизацію, нормалізацію розміру [78, 83, 85]. Рисунок 2.2 ілюструє результат попередньої обробки. Однак розпізнавання МВ пов'язане з 2D-структурою, де порядок штрихів не визначений точно і залежить від уподобань користувача. Тому етап попередньої обробки часто передбачає повторне впорядкування штрихів. Лі та ін. [95] запропонували оптимізований метод X–Y проекцій для впорядкування штрихів, щоб забезпечити незалежність алгориму від їх порядку. Цей метод заснований на виявленні впорядкованих по горизонталі та вертикалі штрихів. Класифікація вертикально упорядкованих штрихів базується на виявленні спеціальних вертикальних символів, таких як дріб, '$\int$','$\sum$' та 'lim'. Цей алгоритм вимагає розпізнавання окремих символів на етапі попередньої обробки.



(а) Початкові точки вхідних штрихів (б) Результат попередньої обробки штрихів

Рис. 2.2. Приклад попередньої обробки рукописних штрихів

Сегментація символів. Для проблем сегментації символів рукописного та друкованого МВ було досліджено багато стратегій. Ранні підходи спирались на техніку проекцій X та Y (X–Y cut) [67, 132, 133] і в основному були спрямовані на сегментацію друкованих МВ. У работах [86, 101, 177] були запропоновані підходи для одночасної сегментації та класифікації символів. Так званий алгоритм “ м’якого рішення” або “soft-decision” намагається знайти найбільш вірогідну послідовність символів і зберігає всі варіанти сегментації та розпізнавання символів до остаточного рішення. В роботі [101] вперше запропонували мережу гіпотез символів (Symbol Hypotheses Net). Завдання сегментації за допомогою прихованих моделей Маркова було запропановано у роботі [86]. Можливість застосування Мінімального Кістякового Дерева (МКД) показана у праці [115]. Автори статті [165] представили метод сегментації, заснований на решітці символів-кандидатів (candidate character lattice) на основі інформації про положення штрихів та математичну структуру, що дало можливість зменшити проблему надмірної та недостатньої сегментації. Методика, що базується на контурних ознаках, була представлена в праці [162]. У дослідженні [75] запропонували метод сегментації символів на основі алгоритму AdaBoost та геометричних багатомасштабних ознаках контексту фігури, який включав 3 групи ознак: пара штрихів, локальне оточення та глобальний контекст фігури. Класифікатор на основі методу опорних векторів для сегментації символів був представлений у праці [97]. Для розрахунку

ймовірності поділу було застосовано дванадцять геометричних ознак та дев'ять додаткових. Остаточне рішення, чи належить пара штрихів одному сегменту, чи ні, було прийнято із застосування порогового значення. В роботі [76] представили «Line-of-Sight» граф штрихів та алгоритм сегментації символів на основі цих графів та «Parzen window-modified Shape Context features». Запропонований набір ознак базується на логаритмо-полярній системі координат і поширений у сфері комп'ютерного зору. Цей перелік робіт можна поповнити результатами досліджень у галузі оптичного розпізнавання символів [52, 129, 155].

Класифікація символів. Зазвичай класифікують три групи методів розпізнавання рукописних символів: онлайн, офлайн та комбіновані [97]. Очевидно, що набір вхідних штрихів може бути перетворений на зображення, а потім для класифікації можна застосовувати офлайн методи. Але також можливо використовувати онлайн методи для розпізнавання в офлайн режимі. Чан [24] розробив алгоритм перетворювання зображення у рукописні штрихи для подальшого застосування онлайн методів розпізнавання МВ. Цей метод включає кілька етапів, такі як бінаризація, скелетування, розкладання на сегменти, реконструкція штрихів, нормалізація порядку тощо. Як зазначає автор, підходи до переходу від офлайн до онлайн вимагають перенавчання онлайн моделі розпізнавання з використанням штрихів, отриманих за допомогою запропонованого алгоритму. У цьому розділі ми зупинимось на онлайн та комбінованих методах. Таблиця 2.1 узагальнює методи класифікації символів.

У наш час sequence-to-sequence методи, засновані на ДРНМ та нейромережевій часовій класифікації (НЧК) [60, 106], демонструють сучасний рівень в задачах обробки послідовностей, таких як розпізнавання мовлення [31] та тексту [128]. Головною особливістю моделей ДРНМ є те, що вони можуть зберігати та використовувати контекстну інформацію протягом тривалого періоду, беручи до уваги як наступні, так і попередні частини вхідної послідовності. НЧК відноситься до функцій оцінки, а застосування НЧК передбачає введення додаткового символу '*Blank*' в алфавіті для забезпечення відокремлення повторюваних символів. НЧК функція втрати $O(S)$ визначається як від'ємна log-імовірність правильного мар-

Таблиця 2.1. Категоризація методів класифікації символів

Категорія	Метод класифікації
Онлайн	Hidden Markov Models [74, 86, 176, 178]
	Bidirectional Long Short-Term Memory [38, 196]
	Bidirectional Long Short-Term Memory and Connectionist Temporal Classification [105]
	Nearest-neighbor classification [153]
	Multi Layer Perceptron [12]
	Adaptive Resonance Theory neural architecture [42]
	Time Delayed Neural Network [15]
	Conditional Random Fields (CRF) [91]
	Artificial Neural Networks and multi-layer perceptron as a junk classifier [13]
	Template Matching [150]
	Hausdorff ARTMAP Neural Network [160]
Комбіновані	Support Vector Machines [83]
	Hidden Markov Models (online + offline features) [4]
	Long Short-Term Memory and Convolutional Neural Network [37]
	Long Short-Term Memory and Hybrid Features set [123]
	Artificial Neural Networks with rejection of false hypotheses [82]
	Markov Random Field and Modified Quadratic Discriminant Functions [97]
	Squeeze-extracted multi-feature convolution neural network [47]

кування навчального набору S, де x – вхідна послідовність, а z – «Ground truth» маркування:

$$O(S) = -\ln\left(\prod_{(x,z)\in S} p(z|x)\right) = -\sum_{(x,z)\in S} \ln p(z|x) \tag{2.1}$$

Однак, навіть незважаючи на те, що НЧК має тенденцію знаходити передбачення символів у відповідній частині вхідної послідовності [61], використання НЧК є проблематичним, оскільки розпізнавання РМВ вимагає сегментації з високою точністю. У роботі [105] було досліджено набір з 25 онайн та псевдо-офлайн ознак та запропоновано підмножину з 16 ознак для систем розпізнавання на основі прихованих моделей Маркова та ДРНМ. Автори [38] вивчали метод розпізнавання на основі ДДКЧП за допомогою 6 ознак, що засновані на часі (x_i, y_i – нормовані

координати; x'_i, y'_i – похідні; d_i – відстань між поточною та наступною точками; $EndPoint_i$ – прапорець останньої точки штриха). У своїй дисертації Жанг [195] запропонувала метод, який використовує вектор ознак із 5 елементів ($\sin\theta_i$, $\cos\theta_i$, $\sin\Delta\theta_i$, $\cos\Delta\theta_i$, $PenUD_i$) та «in-air» точки (невидимі штрихи, що з'єднують два видимих штрихи). Щоб подолати проблему сегментації НЧК, вона запропонувала модифікований алгоритм НЧК, який був названий локальним НЧК. Волкова та ін. [168] запропонували полегшену НМ для корекції помилок сегментації, які можуть виникати у застосуванні НЧК.

ЗНМ поширені в задачах офлайн розпізнавання і відіграють важливу роль у програмах комп'ютерного зору [18, 87, 181]. Також є кілька робіт, які базуються на ЗНМ для розв'язання завдань онлайн-розпізнавання. Так, у роботі [127] автори представили комбінацію ЗНМ та ДКЧП для розпізнавання символів, що значно підвищило якість. У праці [47] було представлено підхід до розпізнавання окремих символів, який називається «squeeze-extracted multi-feature convolution neural network» (SE-MCNN). Цей підхід використовує ознаки у 8 напрямках [17], щоб перетворити шлях штриха в карту об'єктів і тим самим компенсувати втрату динамічної інформації. Для офлайн режиму відповідні ознаки у 8 напрямках генеруються для зображення за допомогою фільтра Габора. Крім того, як заміна softmax була запропонована функція «Joint Loss», щоб модель могла краще розрізнити подібні рукописні символи та зменшити варіації ознак у класі символів.

На відміну від класичного підходу до розпізнавання виразу, автори статті [136] розробили поступовий метод розпізнавання онлайн МВ. Кандидати символів оновлюються після отримання кожного нового штриха. Запропонований метод скоротив час очікування, але цей підхід залежить від порядку штрихів і, отже, має обмеження, пов'язані з обробкою «відстрочених» штрихів.

2.3. Методи структурного аналізу

Структурний аналіз є останнім кроком у класичному робочому процесі розпізнавання МВ. На цьому етапі аналізуються просторові зв'язки між розпізнани-

ми символами та групою символів. І, базуючись на інформації про просторові зв'язки, система генерує найкращу інтерпретацію МВ у двовимірних позначеннях. Існує багато підходів до структурного аналізу, але всі ці методи можна класифікувати на 2 групи: графові та граматичні. Методи, засновані на графах, створюють спрямований граф на основі геометричних особливостей штрихів та/або символів, і часто такі методи передбачають додаткові правила виявлення та виправлення помилок. У свою чергу граматичні підходи спираються на правила виводу граматики та синтаксичний аналіз. Весь розрахунок оцінки найкращої інтерпретації МВ базується на ймовірностях розпізнавання символів, правил граматичного виводу, просторових відношень та особливостях мовної моделі, що використовується для отримання більш точного передбачення. Зазвичай складність підходів, орієнтованих на граматику, вища, але вони перешкоджають побудові неправильного виразу, такого як $Ca + b)$. Більше того, створення правил виводу у граматичних підходах має здійснюватися дуже обережно, беручи до уваги конкретну галузь (наприклад, хімію, геометрію, фізику тощо). Загальна кількість правил виводу може легко перевищувати кілька сотень. Однак, незважаючи на складність, ці підходи поширені та демонструють кращу якість порівняно з графовими. Це підтверджується тим фактом, що використання граматики може виправити помилки в розпізнаванні символів або класифікації просторових відношень.

Класифікація просторових відношень представлена в обох підходах структурного аналізу. Зазвичай виділяють такі класи просторових відношень: '*Right*' (ab), '*Subscript*' (a_b), '*Superscript*' (a^b), '*Above*' ($\overset{a}{\sum}$), '*Under*' ($\underset{a}{\sum}$), '*Inside*' ($\sqrt{a}$). Список просторових класів також може включати: '*Pre-Subscript*' ($_b a$), '*Pre-Superscript*' ($^b a$). Замість створення окремого класу для ступеня кореня ($\sqrt[b]{a}$) часто відокремлюють відношення типу '*Pre-Superscript*' або '*Above*'. Незважаючи на невелику кількість просторових класів відношень, визначення належного типу все ще є складним завданням. Багато в чому це пов'язано з частою плутаниною, що виникає в рукописних МВ між відношенням вправо та верхнім чи нижнім індексами. Дослідники вивчили багато методів класифікації просторових відношень. Короткий огляд наведено в Таблиці 2.2. Існує два типи наборів ознак для класи-

фікації просторових відношень: 1) геометричні, основані на обмежувальних прямокутниках, та 2) фігурні, що беруть до уваги фактичну форму символів. У всіх підходах, що базуються на обмежувальних прямокутниках, друкарські класи символів (див. Рисунок 2.3) використовувались для обчислення геометричних ознак. У статті [97] було представлено концепцію «body box» на основі 4 класів друкарських символів. Підходи, засновані на особливостях фігури, будують контекст фігури як дескриптор у вигляді полярної гістограми. Приклади геометричних та фігурних ознак показані на Рисунку 2.4. Підходи, засновані на геометричних та фігурних ознаках, мають приблизно однакову точність передбачення.

Таблиця 2.2. Підходи до класифікації просторових відношень

Набір ознак	Алгоритм класифікації
Геометричні	Fuzzy membership functions for subscript and superscript analysis [194]
	Support Vector Machines (6 Features) [151]
	Support Vector Machines (9 Features) [5]
	Sequence of Support Vector Machines models (4 Features) [97]
	Bayesian classifier (2 Features) [9]
	Gaussian Mixture Model [15]
	Random Forest ($\approx$ 200 features: Fuzzy Membership + Symbols typology + Geometric Features) [107]
Фігурні	Support Vector Machines for Polar Shape Matrix classification [123]
	Nearest-Neighbor [135]
Комбіновані	Random forest for stroke pair and symbol pair relation classification (8 Geometric + Parzen window modified Shape Context in polar coordinate system) [77]
Інші	Simultaneous character recognition and spatial relation classification with tree-based BLSTM [195]
	Heuristic Rules [156]

Графові методи структурного аналізу. Майже всі методи, засновані на графах, спрямовані на побудову орієнтованого графа, де кожен вузол представляє символ, а кожне ребро – просторове відношення. Такі графи називаються Деревом Компонування Символів (ДКС) або Деревом Відношень Символів. Основною перевагою підходів на основі графів є низька складність, яка зазвичай не перевищує $O(N^2)$, де N – це кількість розпізнаних символів.

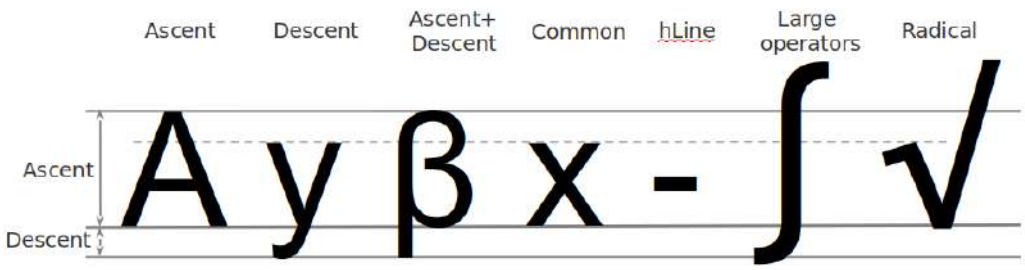
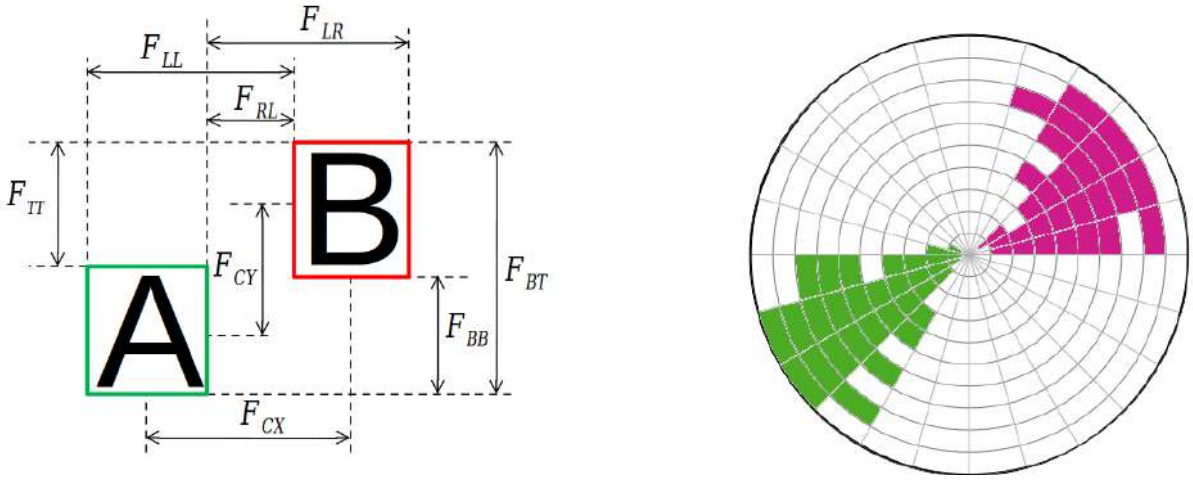


Рис. 2.3. Приклади друкарських класів символів



(а) Геометричні ознаки (б) Фігурні ознаки

Рис. 2.4. Приклади ознак класифікації просторових відношень

У роботі [100] автори запропонували систему побудови виразів за допомогою рекурсивного аналізу та групування символів навколо спеціальних символів, таких як ‘$\sum$’, ‘$\int$’, ‘$\prod$’, ‘$\sqrt{}$’ та знака дробу. Аналіз просторових відношень включає правила блокування нижнього та верхнього індексів для окремих категорій символів. На завершальному етапі виправлення помилки розпізнавання виконується за допомогою набору заздалегідь визначених правил. Мережа віртуальних зв’язків була запропонована у роботі [46]. Ця мережа має кілька ребер з різними мітками просторових відношень та локальною вагою. Процес розпізнавання складається з двох етапів: побудова мережі віртуальних зв’язків та пошук остаточного кістякового дерева. Остаточне дерево визначається мінімальними витратами, які складаються з локальної та глобальної ваги. Локальна вага визначається неоднозначністю рішення щодо типу відношення між двома символами. Глобальна вага відображає глобальну структуру і розраховується на основі набору правил. У

статті [164] була продовжена попередня робота та запропоновано підхід до фільтрації неправильних МВ, використовуючи граматику. У праці [186] автори дослідили підхід до побудови дерева компонування символів шляхом пошуку лінійних структур («базових ліній»). А в роботі [140] вирішували проблему неоднозначності шляхом поступового додавання символу по одному та оцінки кожної гіпотези виразу за допомогою евристичної функції.

Автори статті [156] були першими, хто запропонував метод структурного аналізу МВ за допомогою МКД та домінування символів. Вони застосували алгоритм Прима для побудови ненаправленого МКД. Функція ваги $W(S_T, S_N)$ базується на перевірці домінування під час обчислення ваги ребра між символами. Якщо символ T домінує над символом N, тоді функція W повертає мінімальну відстань між сусідніми точками символів S_T і S_N. В іншому випадку він повертає відстань між центроїдами S_T і S_N. Далі цей метод був розширений у праці [77]. Спочатку графік прямої видимості будується з урахуванням видимості між символами. Класифікатор просторових відношень виступає як вагова функція для генерації оцінок ребер у графіку прямої видимості. Потім алгоритм Едмондса застосовується для пошуку кістякового дерева. Крайній лівий вузол на основній базовій лінії визначається спеціальним фіктивним символом, який був вставлений на графік прямої видимості на початку алгоритму.

Концепція графа нечіткої видимості була представлена у праці [107]. Ребра графа визначаються класифікатором просторових відношень, що включає додатковий клас ‘*Невизначено*’. Для перетворення отриманого графа нечіткої видимості у МВ застосовується аналізатор із заздалегідь визначеними правилами для видалення зайвих ребер. Жанг запропонувала деревовидну архітектуру ДДКЧП [197], яка генерує ДКС безпосередньо з вхідних штрихів і не містить класичних етапів: розпізнавання символів та структурного аналізу. Проміжний граф штрихів будується з використанням просторових відношень та часових ознак. Потім кілька дерев перетворюються за допомогою деревовидної ДДКЧП. Кінцевий результат будується з декількох ДКС на основі найбільшої ймовірності.

Граматичні методи структурного аналізу. Синтаксичні підходи до розпізнавання 2D мов вивчалися протягом останніх п'ятдесяти років [10, 20, 80, 94, 113, 139, 163]. Усі граматичні техніки засновані на формалізмі КВГ. На сьогодні підходи, що засновані на 2D стохастичних контекстно-вільних граматиках (СКВГ), є найбільш вивченими в галузі розпізнавання МВ. У роботі [34] було вперше запропоновано ідею використання 2D–СКВГ для розпізнавання МВ. Запропонований підхід спирається на двовимірну ймовірнісну версію алгоритму Кокке-Янгера-Касамі (КЯК). Автори статті [183] запропонували систему, яка одночасно розпізнає символи та аналізує структуру. Інакше кажучи, була запропонована граматика на основі штрихів. Далі Альваро вивчав та розробляв метод 2D–СКВГ у численних статтях [2, 5, 6, 123]. Він описав підхід на рівні штрихів, заснований на синтаксичному аналізі 2D ймовірнісних КВГ з використанням алгоритму КЯК. Зазвичай алгоритм розбору КЯК ілюструється у вигляді таблиці (Рисунок 2.5), де кожна клітинка містить гіпотези кандидата та їх оцінку. Для граматики на основі штрихів нижній рівень містить гіпотези для кожного штриха замість символу. Правила виводу для алгоритму КЯК визначені в нормальній формі Хомського (НФЧ):

$$\begin{aligned} A &\Longrightarrow \alpha \\ A &\Longrightarrow BC, \end{aligned}$$

де A, B і C – нетермінальні символи, α – термінальний символ.

Приклади правил виводу алгоритму синтаксичного аналізу МВ:

$$\begin{aligned} TERM &\Longrightarrow symbol \\ TERM &\Longrightarrow number \\ LEFT &\overset{\rightarrow}{\Longrightarrow} operator\ TERM \\ EXPR &\overset{\rightarrow}{\Longrightarrow} TERM\ LEFT \\ DENOM &\overset{\downarrow}{\Longrightarrow} hline\ TERM \\ FRAC &\overset{\downarrow}{\Longrightarrow} TERM\ DENOM \end{aligned}$$

Кілька досліджень [23, 98, 104] стосувалися ефективності та складності алгоритму КЯК, який дорівнює $O(N^3|P|)$ для одновимірних мов та $O(N^4|P|)$ для

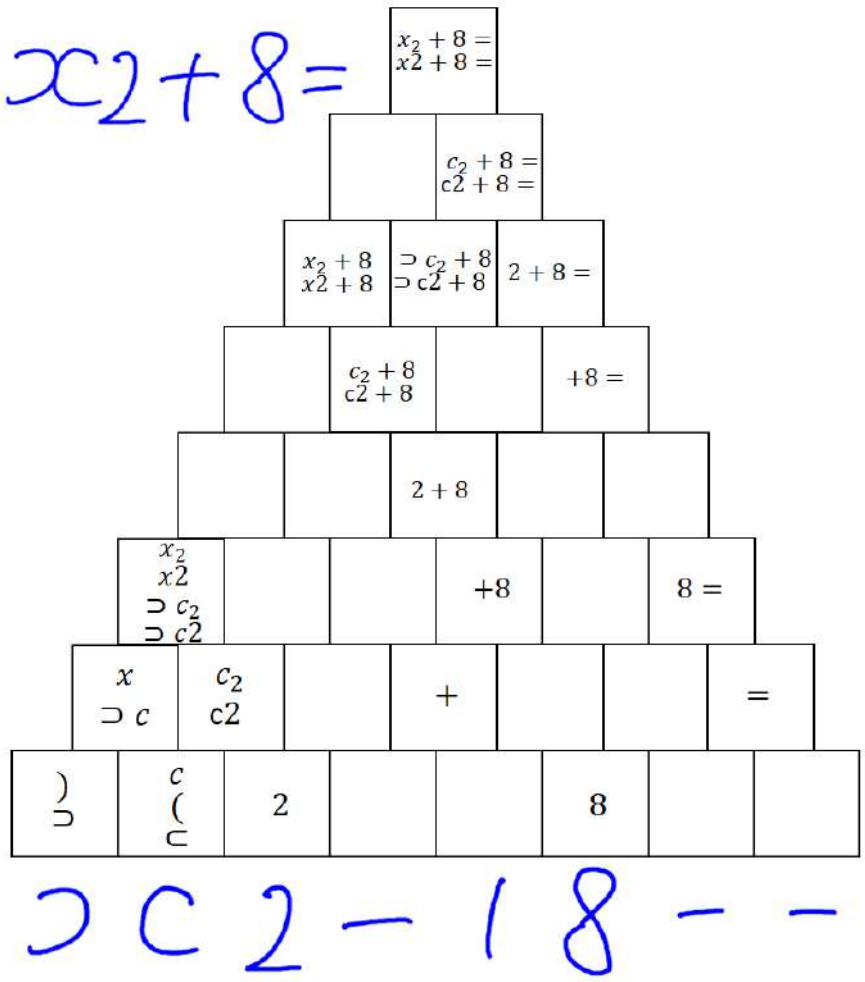


Рис. 2.5. Приклад таблиці Кокке-Янгера-Касамі для виразу ‘$x_2 + 8 =$’

2D-мов, де $|P|$ – кількість правил виводу в граматиці, а N – кількість примітивів символів(штрихів). У своїх роботах Альваро представив підходи до зменшення складності і, нарешті, досяг $O(N^3 \log N|P|)$ [6]. У праці [98] автори запропонували методи скорочення кількості гіпотез в дереві синтаксичного аналізу за рахунок часткового видалення малоймовірних гіпотез.

Нечітку реляційну контекстно-вільну граматику можна відрізнити від інших підходів за способом побудови граматики. Цей підхід, запроваджений у роботі [108], і має меншу обчислювальну складність, ніж алгоритм КЯК. Складність нечіткої реляційної КВГ складає $O(N^{3+K}|P|K)$, де K – кількість токенів правої сторони у найбільшому виводі. Ще однією особливістю цього підходу є більш виразна граматика, яка дає можливість уникнути створення сотень правил виводу.

2.4. Наскрізні рішення на основі нейронних мереж

В останні роки наскрізні системи розпізнавання, що базуються на автокодувальниках, є домінуючими в розпізнаванні мовлення, автоматичній анотації зображень, машинному перекладі. Ці системи також називаються керованими даними, що означає відсутність чітко визначених знань домену. У цьому розділі ми розглянемо наскрізні рішення, засновані на автокодувальниках, які зараз в основному застосовуються для розпізнавання РМВ в режимі офлайн, оскільки вони демонструють вражаючі результати, і деякі методи можна застосувати до онлайн розпізнавання. Рисунок 2.6 ілюструє загальне представлення автокодувальника з механізмом уваги. Кодувальник перетворює вхідну послідовність у приховане представлення, а потім декодувальник за допомогою механізму уваги відображає приховане представлення у МВ.

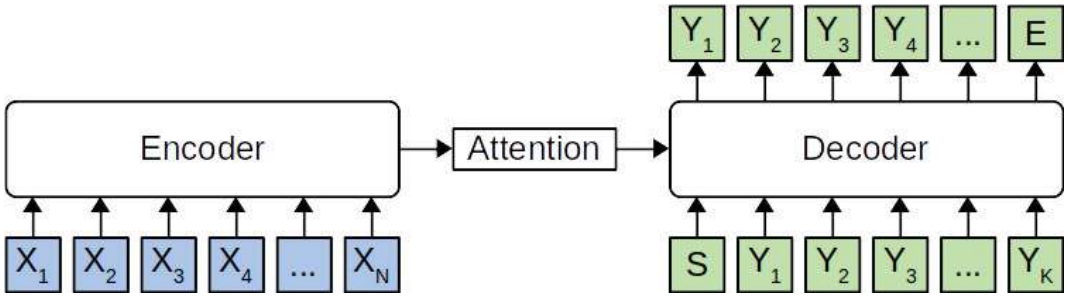


Рис. 2.6. Структура автокодувальника «Encoder-Attention-Decoder»

Жанг провів кілька експериментів з різними архітектурами НМ. У праці [193] представлено модель НМ «Watch, Attend та Parse» (WAP) для офлайн розпізнавання МВ. Запропонована система використовує архітектуру VGG для реалізації кодувальника і вентильний рекурентний вузол (ВРВ) для декодувальника. Модель уваги змушує декодувальник зосередитись на певній частині вхідного зображення, щоб розпізнати один символ або просторовий зв’язок між символами. Пізніше вони вдосконалили рішення [191], замінивши архітектуру кодувальника на DenseNet та застосувавши модель багатомасштабної уваги, що дало змогу системі краще мати справу з різними розмірами рукописних символів. Його дослідження не обмежуються лише розпізнаванням в офлайн режимі. У роботі [190] було запропоновано архітектуру автокодувальника на основі ВРВ для систем онлайн-

розпізнавання. Він складається з одношарового декодувальника із вбудованим механізмом уваги на основі покриття та багатошарового двонаправленого ВРВ кодувальника. Трохи пізніше вони вдосконалили своє рішення та запропонували структуру «Track, Attend та Parse» (TAP) [192], де в декодувальник було інтегровано керовану гібридну увагу, що складається з механізму просторової уваги на основі охоплення, механізму часової уваги та керівника уваги. Керівник уваги потрібен під час навчального процесу як метод регуляризації механізму просторової уваги, а механізм тимчасової уваги відповідає за баланс між трекером та мовною моделлю. Нарешті, TAP інтегрувався з WAP та мовною моделлю на основі ВРВ у процедуру променевого пошуку, де мовна модель на базі ВРВ навчалася на додатковому текстовому наборі даних.

У статті [68] була представлена структура наскрізної ЗНМ для офлайн розпізнавання друкованих МВ. Різні механізми уваги для архітектури автокодувальника в офлайн розпізнаванні друкованих МВ були досліджені у праці [41]. У роботі [99] було запропоновано наскрізне рішення з 3 шарами: ЗНМ як екстрактор ознак, ДДКЧП модуль як кодувальник та ДКЧП модель на основі уваги як декодувальник для генерації LaTeX. Автор статті [196] запропонував рішення, засноване на одній ДКЧП НМ, яка генерує вихід у вигляді одновимірної послідовності, що описує граф із просторовими відношеннями. Усі представлені підходи та моделі пройшли навчання та оцінку на відкритих наборах даних CROHME. Остаточні результати для підходу TAP + WAP + LM [192] були отримані за допомогою ансамблю із трьох екземплярів кожної моделі.

Автори статті [172] адаптували вищезазначені методи і зосередилися на розробці підходів, заснованих на архітектурах кодувальника-декодувальника, що поєднують переваги онлайн і офлайн представлення МВ. Вони запропонували «Multi–Modal Attentional Network» (MAN) [172] та «Stroke Constrained Attentional Network» (SCAN) [171]. Архітектура MAN складається з ЗНМ та стеку двонаправлених ВРВ для кодування онлайн каналу вхідних даних та DenseNet [79] для кодування офлайн-каналу. Декодувальник складається з двох односпрямованих ВРВ з мультимодальним механізмом уваги. Порівняно з підходом MAN, архітектура

SCAN (Рисунок 2.7) використовує маски штрихів для вирівнювання між онлайн- та офлайн-каналами. Кожна онлайн-маска штриха містить інформацію про те, чи належить точка i до штриха j, чи ні, а офлайн-маска штриха визначає, чи (x, y) піксель належить j штриху. Застосування таких масок забезпечує злиття різних каналів у декодувальнику.

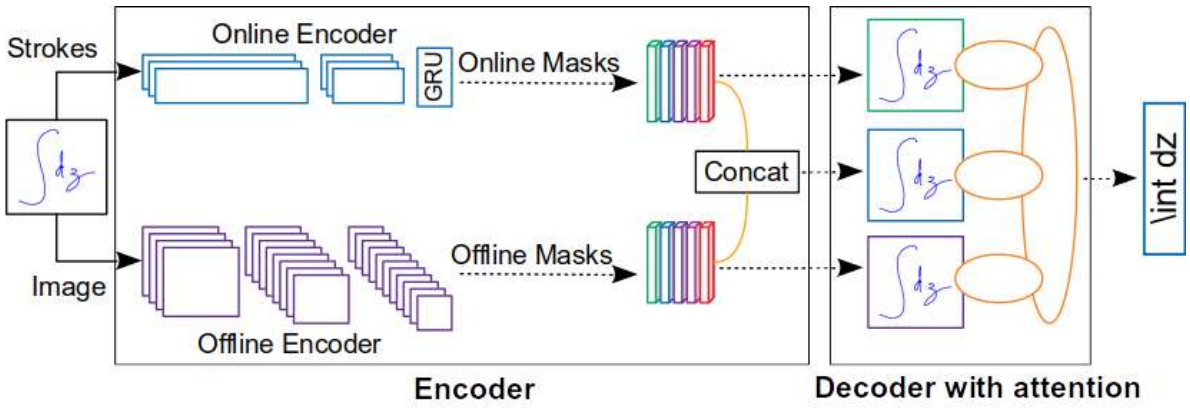


Рис. 2.7. Архітектура «stroke constrained attentional network«(SCAN)

У роботі [44] автор запропонував «dual loss attention network» для поліпшення розпізнавання РМВ в умовах нестачі даних для навчання за допомогою друкованих МВ, які є результатом рендерінгу з існуючого LaTeX корпусу. Так, окрім функції втрати декодувальника, запропонований механізм уваги використовує функцію втрати відповідності контексту, що забезпечує вивчення семантичних особливостей кодувальника та граматики LaTeX. У роботі [103] були запропоновані методи, що вирішують проблему різного розміру символів. По-перше, було запропоновано збільшення кількості зразків, застосовуючи різні коефіцієнти масштабування замість нормалізації до одного розміру. По-друге, автори запровадили «drop attention» механізм, який спрямований на покращення точності за рахунок тренування моделі розпізнавання в умовах, коли механізм уваги декодувальника не є точним.

У Таблиці 2.3 наведено оцінку точності розпізнавання та кількість ваг у запропонованих наскрізних рішеннях.

Таблиця 2.3. Порівняння наскрізних рішень

Архітектура НМ	Кількість ваг	CROHME		
		2014	2016	2019
BLSTM [196]	≈ 300 000	30.57%	–	–
WAP [193]	≈ 1 400 000	46.55%	44.55%	–
WAP + MHA [43]	–	–	56.76%	59.72%
WAP + MHA + Stacked decoder [43]	≈ 7 160 000	–	57.72%	61.38%
GRU [190]	–	52.43%	–	–
MSA [191]	≈ 6 200 000	52.80%	50.10%	–
TAP [192]	≈ 5 700 000	55.37%	50.22%	–
TAP+WAP [192]	≈ 11 100 000	60.34%	55.27%	–
TAP+WAP+LM [192]	–	61.16%	57.02%	–
PAL [180]	–	47.06%	–	–
Watch Step by Step [170]	–	49.46%	–	–
MAN [172]	–	52.43%	49.87%	–
Enhanced MAN [172]	–	54.05%	50.86%	–
Online SCAN [171]	–	51.22%	46.12%	46.49%
Online SCAN + TAP [171]	–	52.64%	47.17%	47.62%
Offline SCAN [171]	–	47.67%	46.64%	47.62%
Offline SCAN + WAP [171]	–	49.39%	49.60%	49.62%
Online + Offline SCAN (decoder fusion) [171]	–	55.38%	52.22%	53.88%
Online + Offline SCAN (encoder fusion) [171]	–	57.20%	53.97%	56.21%
Dual loss attention network [44]	–	51.88%	51.53%	–
Scale augmentation and drop attention [103]	–	60.45%	58.06%	–
Residual birnn [72]	–	54.56%	–	–
QD-GGA [110, 111]	–	32.04%	32.84%	40.65%
Recursive Recognition [36]	≈ 7 400 000	–	–	58.7%
Stroke Based Posterior Attention [179]	–	54.26%	51.75%	–
Symbol-relation temporal classifier [166]	–	44.12%	41.76%	–
Bidirectionally Trained TRansformer [198]	–	53.96%	52.31%	52.96%
ABM [21]	–	56.85%	52.92%	53.96%

2.5. Висновки до Розділу 2

В розділі проведений аналіз існуючих підходів та представлені результати найбільш цікавих досліджень. Були розглянуті основні етапи розвитку систем розпізнавання МВ та етапи процесу розпізнавання. Більшість публікацій за останні кілька років пов’язані з різними архітектурами НМ та методами підготовки даних для навчання таких систем. Тому найбільш перспективним вектором розвитку є системи, засновані на наскрізному навчанні. Такі системи демонструють перевагу в точності розпізнавання. Підтвердженням цього є результати відкритого змагання CROHME 2019 [112] з розпізнавання РМВ, в якому перемогла система, побудована на об’єднанні кількох НМ [192] для онлайн та офлайн модальностей.

Ця система продемонструвала точність розпізнавання на рівні 80.73%. Але такі системи мають величезну складність, що не дає можливості використовувати їх за умови обмеження обчислювальних ресурсів. Тому їх застосування на даний момент часу обмежена хмарними обчисленнями.

Послідовні рішення в цей час не застосовуються, оскільки мають ряд недоліків. Основний із них – це накопичення помилок у переходах між різними етапами виконання. Інтегровані системи ще не повною мірою розкрили свій потенціал. За останні кілька років якість розпізнавання таких систем виросла більш ніж на 10% (67.65% у 2016 році [121] та 79.82% у 2019 році [112]). Порівняно з наскрізними рішеннями, інтегровані методи мають приблизно один відсоток відставання в точності розпізнання, але при цьому вони менш вимогливі до обчислювальних ресурсів.

В інтегрованих системах виділяють два основних напрямки: графові та граматичні. Перша група методів забезпечує побудову виразу за рахунок геометричних особливостей і їх алгоритмічна складність не перевищує $O(N^2)$. На відміну від першої групи, граматичні методи спираються на математичний формалізм КВГ. Такі методи були запозичені з галузі обробки природних мов, де обробка пов'язана з одновимірною структурою. Однак складність таких алгоритмів значно вища і становить $O(N^3 \log N|P|)$, що в деяких випадках може призводити до помилки часу виконання.

Розглядувана галузь досліджень викликає постійний інтерес інженерів і вчених, що підтверджується кількістю наукових публікацій (Рисунок 2.8).

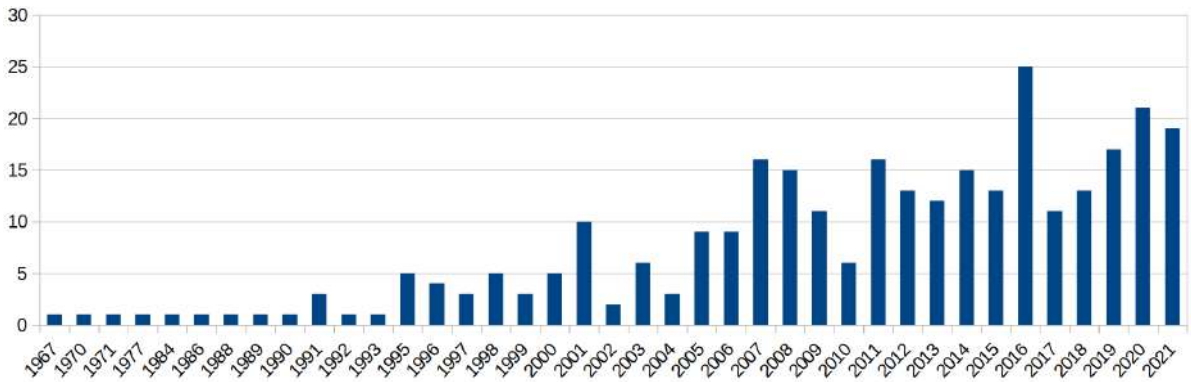


Рис. 2.8. Кількість публікацій за темою розпізнавання РМВ по роках

РОЗДІЛ 3

МЕТОД СЕГМЕНТАЦІЇ ТА КЛАСИФІКАЦІЇ СИМВОЛІВ НА ОСНОВІ РЕКУРЕНТНИХ НЕЙРОННИХ МЕРЕЖ

Розділ присвячений проблемі сегментації та класифікації символів. Цей етап часто називають маркуванням послідовності (*sequence labeling*). У машинному навчанні маркування послідовностей – це завдання розпізнавання зразків (*pattern recognition*), що передбачає присвоєння мітки кожному члену вхідної послідовності. Прикладом завдання маркування послідовності є розмічування частин мови, де потрібно визначити частину мови для кожного слова у вхідному реченні.

Сегментацію та класифікацію символів пропонується здійснювати на основі архітектури РНМ, яку називають мережею двонаправленої довгої короткочасної пам’яті. Для тренування використовується нейромережева часова класифікація для знаходження таких параметрів РНМ, які максимізують ймовірність послідовностей міток у навчальному наборі зразків. Секції 3.1 та 3.2 містять опис ДДКЧП архітектури та нейромережевої часової класифікації. Матеріали цих секцій основані на роботі Алекса Грейвса [60] та подають стислий її опис для розуміння дисертаційної роботи в цілому. Особливості попередньої обробки вхідних жестів для 2D-мов розглянуті у Секції 3.3. Загальна архітектура НМ наведена у Секції 3.4. В Секції 3.5 розглянуто основні недоліки, пов’язані із застосуванням НЧК, та запропоновано метод для підвищення точності сегментації та класифікації символів за допомогою додаткової легковагової НМ.

3.1. Двонаправлена довга короткочасна пам’ять

Вперше мережу довгої короткочасної пам’яті (ДКЧП) було запропановано Хохрайтером та Шмідгубером у 1997 році [71]. В тому ж році Шустером та Паліваломбула винайдена двонаправлена РНМ [147], яка використовує скінченну послі-

довність, щоб передбачувати або маркувати кожен елемент цієї послідовності на основі як минулого, так і майбутнього контексту цього елементу. Двонаправлена довга короткочасна пам'ять (ДДКЧП) – це архітектура нейроних мереж, що наразі застосовується для вирішення багатьох завдань, таких як прогнозування часових рядів [146], розпізнавання мовлення [62, 63], навчання граматики [55, 145], розпізнавання людських дій [16], розпізнавання рукописного вводу [106].

Розгорнута РНМ зображена на Рисунку 3.1. На кожному кроці часу t вхідні дані – це вектор ознак, а вихід – це вектор ймовірностей для різних класів. Поточний вхід надходить до поточного прихованого шару завдяки з'єднанням від вхідного шару до прихованого шару, яким призначено вагу '$w1$'. З'єднання, яким призначено вагу '$w2$', відповідають за передачу інформації між прихованим шаром у момент часу $t-1$ до прихованого шару в момент часу t. Потік збудження з прихованого шару до вихідного шару забезпечується з'єднаннями з вагою '$w3$'. Зверніть увагу, що '$w1$', '$w2$' та '$w3$' представляють вектори ваг замість окремих значень ваги, і вони використовуються повторно для кожного кроку часу.

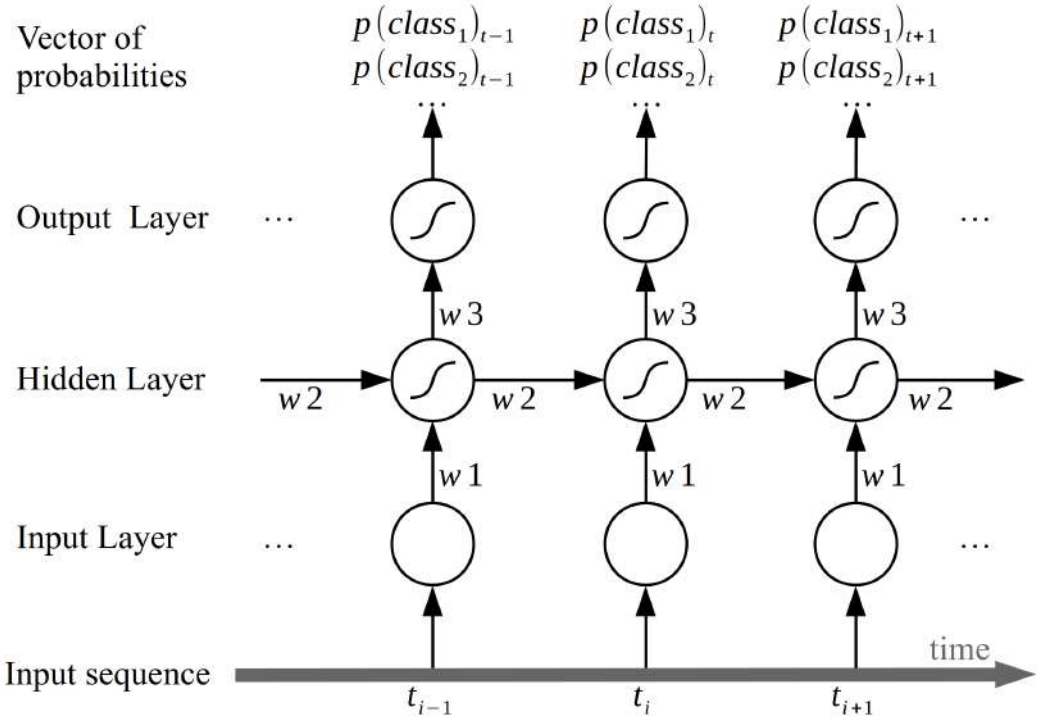


Рис. 3.1. Розгорнута рекурентна нейронна мережа

Насправді, для задачі маркування послідовностей майбутня інформація також важлива як і попередній контекст. Двонаправлена РНМ (Рисунок 3.2) здатна вра-

ховувати контекстну інформацію як з минулого, так і майбутнього. Представлена НМ має 2 окремі приховані шари для прямого та зворотного напрямку. Кожен із них обробляє вхідну послідовність в одному напрямку, та не існує жодних зв'язків між цими прихованими шарами. Також ці два шари з'єднані з одним вихідним шаром. Така двонаправлена структура НМ здатна використовувати минулий і майбутній контекст для розпізнавання кожної точки вхідної послідовності.

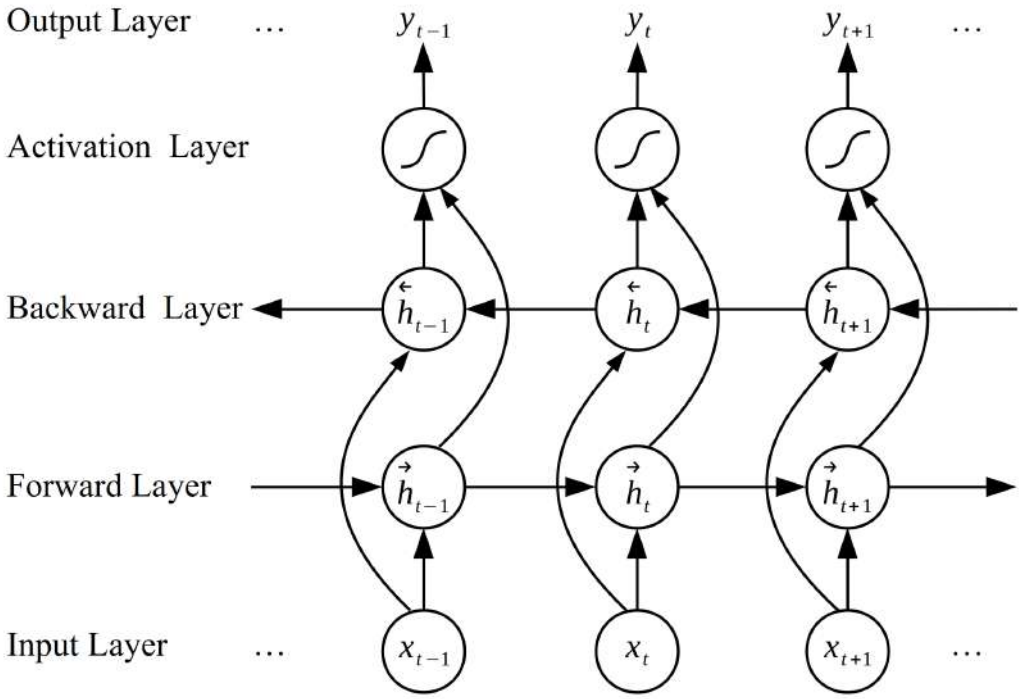


Рис. 3.2. Розгорнута двонаправлена рекурентна нейронна мережа

Нейронна мережа ДКЧП складається з шарів, які містять вузли ДКЧП. Кожен вузол ДКЧП має декілька вентилів (*gate*), які керують потоком інформації. Існує кілька варіантів реалізації вузлів ДКЧП. У перших реалізаціях ДКЧП містив лише вхідні та вихідні вентилі. Забувальний вентиль [56] та ваги вічків (*peephole weights*) [57], що з'єднують вентилі з коміркою пам'яті, були додані пізніше. Вузол ДКЧП здатний запам'ятовувати значення для довгих або коротких проміжків часу [204]. У цій роботі використовується стандартний РНМ вузол з блоком пам'яті. Такий вузол має три вентилі (gates) та одну або кілька комірок у блоці пам'яті. Рисунок 3.3 показує РНМ вузол з блоком пам'яті на одну комірку.

Вхідний вентиль контролює міру входження нового значення до пам'яті. Забувальний вентиль контролює міру того, на скільки значення залишається в пам'яті.

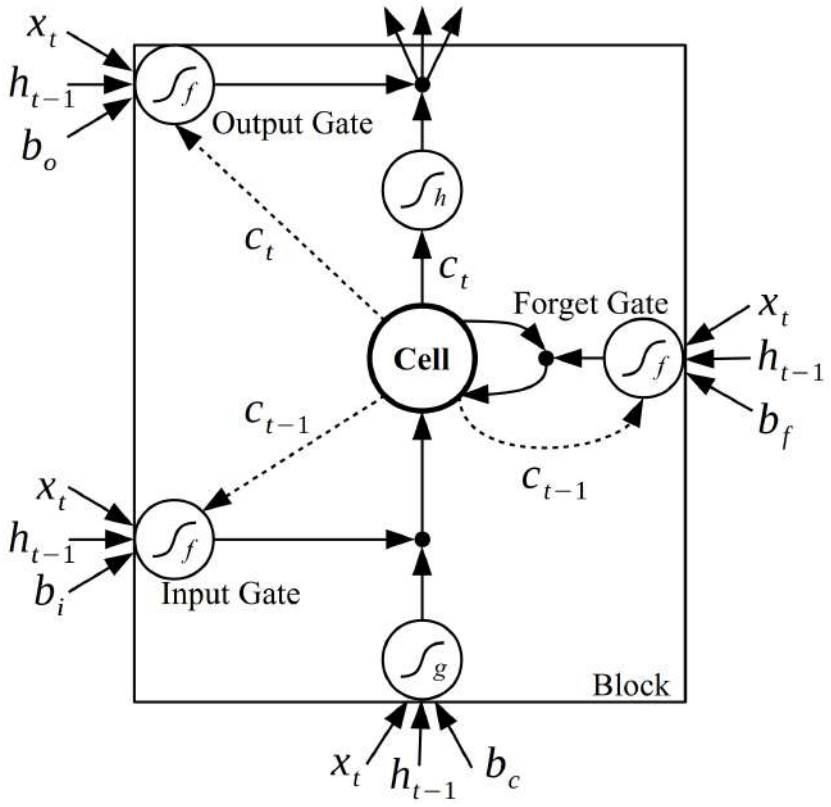


Рис. 3.3. Вузол ДДКЧП з блоком пам’яті на одну комірку

А вихідний вентиль контролює те, на скільки значення в пам’яті використовується для обчислення активації виходу блоку. Далі перелічені вхідні дані, що надходять у три вентилі в момент часу t:

– *Вхідний вентиль (Input gate)*: поточний вхід, активація прихованого шару в момент часу $t-1$, стан блоку пам’яті в момент часу $t-1$.

– *Забувальний вентиль (Forget gate)*: поточний вхід, активація прихованого шару в момент часу $t-1$, стан блоку пам’яті в момент часу $t-1$.

– *Вихідний вентиль (Output gate)*: поточний вхід, активація прихованого шару в момент часу $t-1$, поточний стан блоку пам’яті.

З’єднання, що показані пунктирними лініями від комірки пам’яті до трьох вентилів, називаються вічковими з’єднаннями, та є зваженими з’єднаннями всередині блоку пам’яті. Завдяки цим трьом з’єднанням стан блоку пам’яті доступний усім вентилям. Ці три вентилі підсумовують інформацію зсередини та зовні блоку з різним значенням ваг, а потім застосовують функцію активації f вентиля. Функція активації вентиля, як правило, представлена у вигляді логістичної сигмоїди.

Таким чином, активація воріт знаходиться в діапазоні між 0 (ворота закриті) та 1 (ворота відкриті). Три вентилі керують вузлом шляхом множення, яке на рисунку зображено у вигляді маленьких чорних кіл:

– *Вхідний вентиль (Input gate)*: активація вхідного вентиля відповідає за кількість інформації, яку вузол може отримати від поточного входу. Так, 0 означає, що ніякої інформації вузол отримувати не буде, а значення 1 – що уся отримана інформація буде врахована.

– *Забувальний вентиль (Forget gate)*: активація забувального вентиля відповідає за значення кількості інформації, яку вузол має зберігати з попереднього стану (0 – забути всю інформацію, 1 – запам'ятати все).

– *Вихідний вентиль (Output gate)*: активація вихідного вентиля контролює, скільки інформації з поточного стану буде отримано на виході вузла (0 – поточний стан не впливає на вихід, 1 – весь контекст поточного стану враховується).

Будемо використовувати такі позначення: x_t вхід у момент часу t; i_t та f_t є векторами вхідного та забувального вентилів у момент часу t; c_t – вектор активації комірки; o_t – вихідний вектор комірки; h_t – вихід з блоку на решту мережі; W позначають вагові матриці та b – вектори зміщення. Функція активації σ є логістичною сигмоїдою, а τ – гіперболічний тангенс. Всі початкові значення дорівнюють 0.

Тоді ДДКЧП модель визначається такими рівняннями:

Input Gates:

$$i_t = \sigma_i(x_t * W_{xi} + h_{t-1} * W_{hi} + c_{t-1} * W_{ci} + b_i) \tag{3.1}$$

Forget Gates:

$$f_t = \sigma_f(x_t * W_{xf} + h_{t-1} * W_{hf} + c_{t-1} * W_{cf} + b_f) \tag{3.2}$$

Cells:

$$c_t = f_t c_{t-1} + i_t \tau_c (x_t * W_{xc} + h_{t-1} * W_{hc} + b_c) \tag{3.3}$$

Output Gates:

$$o_t = \sigma_o (x_t * W_{xo} + h_{t-1} * W_{ho} + c_t * W_{co} + b_o) \tag{3.4}$$

Cell Outputs:

$$h_t = o_t \tau_h (c_t) \tag{3.5}$$

Три вентилі дають можливість вузлу вибірково приймати, запам'ятовувати та виводити інформацію, тим самим значно зменшуючи проблему зникнення градієнта. Таким чином, ДКЧП – це система глибинного навчання, яка, на відміну від традиційних РНМ, не має проблеми зникання градієнта. Для прикладу, вузол може повністю запам'ятати вхідні дані в першій точці ($t = 1$), доки забувальний вентиль відкритий та вхідний вентиль закритий на наступних етапах. Тобто ДКЧП може застосовуватися для задач "дуже глибокого навчання" (very deep learning), що потребують спогадів про події, які сталися тисячі кроків тому [204]. Вічкові з'єднання покращили здатність ДКЧП вивчати послідовності, які вимагають точного хронометражу та підрахунку внутрішніх станів [195]. Можливості пам'яті використовуються в багатьох задачах маркування послідовностей.

Здатність ДДКЧП враховувати контекст у двох напрямках дає змогу НМ запам'ятовувати не тільки особливості написання окремих символів, а й приймати рішення про правильне їх маркування, враховуючи геометричні та семантичні особливості сусідніх символів. Для прикладу, практично неможливо відрізнити написання великої і маленької літери c, не маючи інформації про сусідні символи. Іншим прикладом може бути кружок, який залежно від контексту може позначати цифру 0, маленьку та велику літери O, оператор композиції $\circ$ або символ градусу $^\circ$. Рисунок 3.4 містить приклади зображення з абсолютно однаковим кружком, який використовується в різних контекстах та позначає різні математичні символи.

(а) $a \circ b$ (б) 10 (в) 90° (г) ob

Рис. 3.4. Приклад однакового написання символу кружка для позначення різних математичних символів

Наведені приклади дуже прості, та в усіх цих випадках НМ забезпечує правильну сегментацію та класифікацію символів з урахуванням геометричних і семантичних особливостей сусідніх символів. Це забезпечується за рахунок можливості ДДКЧП отримати доступ до контекстної інформації з двох напрямків. Однак існує проблема, що діапазон контексту, до якого можна отримати доступ на практиці, досить обмежений.

3.2. Нейромережева часова класифікація

Задачу тренування нейронних мереж на навчальному наборі найчастіше формулюють як задачу мінімізації функції втрат. У задачах класифікації тренувальний набір складається з пар (x, y), де x – вхідний елемент з описом об'єкта, y – відповідний номер або назва класу. Задачі маркування послідовностей суттєво відрізняються від задач класифікації тим, що вхідними даними є послідовності x, тоді як цілями є послідовність y дискретних міток, що належать кінцевому алфавіту L. В багатьох задачах маркування послідовностей та аналізу часових рядів для тренування використовуються набори, в яких відомо значення істини для кожного часового кроку. Але створення такого набору даних на практиці майже неможливе, бо це вимагає ручної анотації кожної точки вхідного штриха.

У роботі [60] Алекса Грейвса був запропонований метод нейромережевої часової класифікації. Запропонований метод допомагає уникнути дуже ретельної анотації тренувальних наборів даних та добре зарекомендував себе для вирішення

завдань маркування послідовностей при розпізнаванні мовлення [143] та рукописного тексту [116].

Для навчання НМ за допомогою функції втрат НЧК подається вхідна матриця ознак та відповідний вектор символів без прив'язки кожного символу до конкретного моменту часу. Для кодування повторюваних символів додається додатковий символ *blank*. Тоді новий алфавіт L' дорівнює $L \cup blank$. Додатковий символ забезпечує кодування, під час якого ми можемо вставити довільну кількість *blank* символів у будь-яку позицію, які будуть видалені під час декодування D. Однак ми маємо вставити *blank* символ між повторюваними символами. Крім того, кожен символ може повторюватися скільки завгодно разів. Приклади декодування послідовності символів (*blank* позначений символом $-$):

- $D(--1\ 1--1-+----2--2\ 2) = 1\ 1+2\ 2$
- $D(--1--1\ 1\ 1+--2-----2) = 1\ 1+2\ 2$
- $D(-1\ 1-1\ 1\ 1-+2\ 2\ 2\ 2-2\ 2\ 2) = 1\ 1+2\ 2$
- $D(\ 1---1----+2-----2--) = 1\ 1+2\ 2$

Як бачимо, ця схема дає можливість легко створювати різні вирівнювання однієї послідовності символів. НМ навчається виводити закодовану послідовність символів. Активація y_k^t вихідного вузла k в момент часу t – це ймовірність спостереження мітки $k \in L'$ в момент часу t для вхідної послідовності x довжиною T з навчального набору S. Нехай L'^T позначає множину послідовностей над L' довжиною T, і будь-яка послідовність $\pi \in L'^T$ називається шляхом. Тоді ймовірність послідовності π визначається формулою:

$$p(\pi|x,S) = \prod_{t=1}^{T} y_{\pi_t}^t \tag{3.6}$$

Наступним кроком є визначення ймовірності певного маркування $l \in L^{\leqslant T}$, де $L^{\leqslant T}$ це набір можливих маркувань, який визначається функцією декодування $D : L'^T \to L^{\leqslant T}$. Оскільки шляхи взаємовиключні, умовну ймовірність деякого маркування l можна обчислити шляхом підсумовування ймовірностей усіх шляхів, для яких результат декодування однаковий:

$$p(l|x) = \sum_{\pi \in D^{-1}(l)} p(\pi|x) \tag{3.7}$$

Цільова функція втрати НЧК $O(S)$ визначається як негативна ймовірність правильного маркування всіх навчальних прикладів у навчальному наборі S та задається такою формулою:

$$O(S) = -\ln \left(\prod_{(x,z) \in S} p(z|x) \right) = - \sum_{(x,z) \in S} \ln p(z|x) \tag{3.8}$$

де z – це очікуваний результат вхідної послідовності x.

3.3. Попередня обробка вхідних штрихів

Головною метою попередньої обробки є уніфікація вхідної послідовності для зменшення обсягу інформації, усунення нерелевантної інформації та нормалізації почерку. Як правило, нерелевантна інформація включає дублюючі точки, гачки, шум тощо. Підходи до попередньої обробки або нормалізації рукописного тексту добре представлені в наукових роботах [65, 78] і не змінюються вже тривалий час.

Усунення гачків. Зазвичай гачки виникають на початку та/або в кінці штриха. Крім того, гачки, як правило, характеризуються малою довжиною та різкими змінами кута.

Згладжування даних. Цей крок застосовується для зменшення шуму. Найчастіше згладжування даних здійснюється через інтерполяцію кубічними сплайнами. При цьому проводиться пошук гострих кутів (точок), які виступають як контрольні точки.

Нормалізація. Метою нормалізації є усунення деяких варіацій стилів почерку та спрощення форми символів. Зазвичай цей етап включає такі процедури: визначення базової лінії, масштабування та виправлення нахилу.

Передискретизація. Набір вхідних точок кожного штриха змінюється таким чином, щоб точки розташовувалися рівномірно, але при цьому необхідно не втра-

тити важливої інформації. Сучасні пристрої дають можливість зчитувати точки з сенсорного екрана з великою частотою. Це може призводити до того, що один простий жест здатний містити десятки або навіть сотні точок. Також кількість точок залежить від темпу написання МВ. Використання всіх вхідних точок може сильно впливати на здатність правильно запам'ятовувати контекст, якість маркування вхідної послідовності та ефективність розпізнавання МВ, обчислювальна складність якого лінійно залежить від довжини вхідної послідовності.

Переупорядкування штрихів. У розпізнаванні рукописного тексту переупорядкування штрихів пов'язують з так званими «відстроченими» штрихами, коли штрих відокремлений від основного тіла символу, якому він належить, одним або кількома штрихами [159]. Підхід з ДДКЧП та НЧК вимагає на етапі навчання суворої відповідності між порядком символів у вхідній послідовності та відповідною їм послідовністю вхідних жестів.

Процес попередньої обробки вхідних даних для рукописних МВ дещо відрізняється від процесу для рукописного тексту. Це пов'язано з двомірною структурою МВ, де існує кілька базових ліній для кожного підвиразу, та наявністю «подовжуваних» (stretchable) символів. «Подовжуваними» будемо називати символи, розмір яких залежить від внутрішніх елементів, до яких застосовується цей символ. Прикладом таких символів є знаки дужок, знак дробу, стрілочки. Все це призводить до того, що один символ може зустрічатися в виразах у різних розмірах. Більше того, порядок написання елементів виразу може сильно відрізнятися залежно від уподобань користувача. Так, якщо в рукописному тексті порядок не змінюється і залежить від мови (зліва направо або справа наліво), то у написанні МВ порядок штрихів не визначено. Для прикладу, часто дужки розставляються після написання виразу, знак дробу може бути написаний як на самому початку, так і в самому кінці.

Етапи попередньої обробки штрихів для РМВ зображені на Рисунку 3.5. У цій секції ми не будемо зупинятися на кожному етапі попередньої обробки. Детальна інформація може бути знайдена в роботі [78]. Зазначимо лише основні відмінно-

сті, пов'язані зі змінами, які внесені в процес попередньої обробки РМВ порівняно з обробкою рукописного тексту.

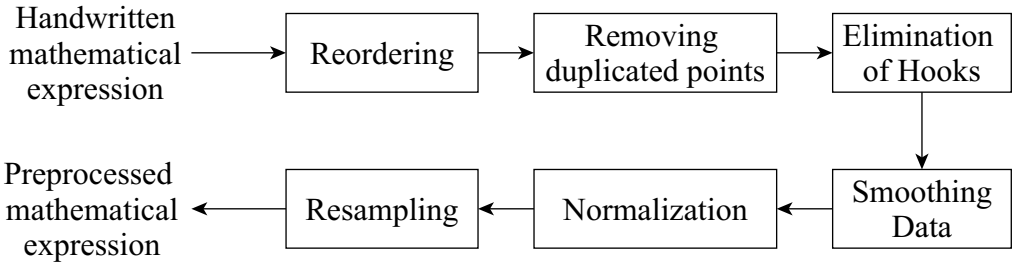
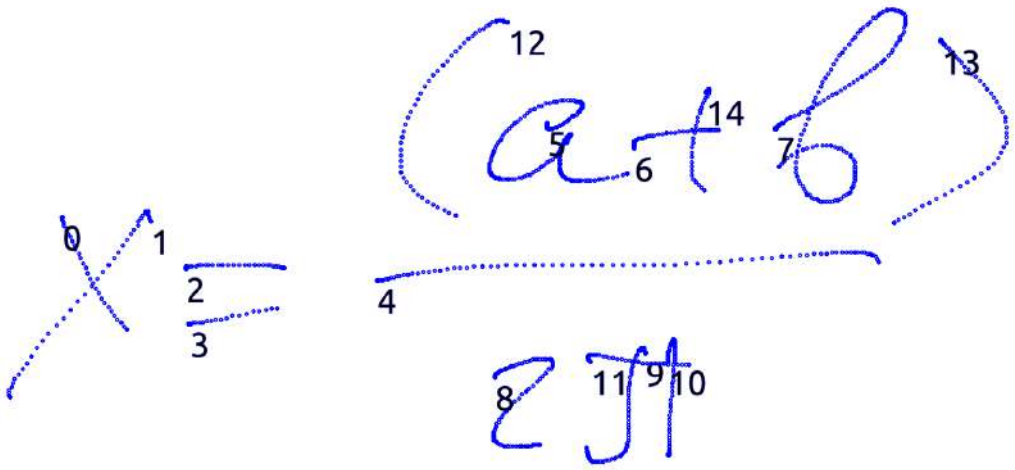


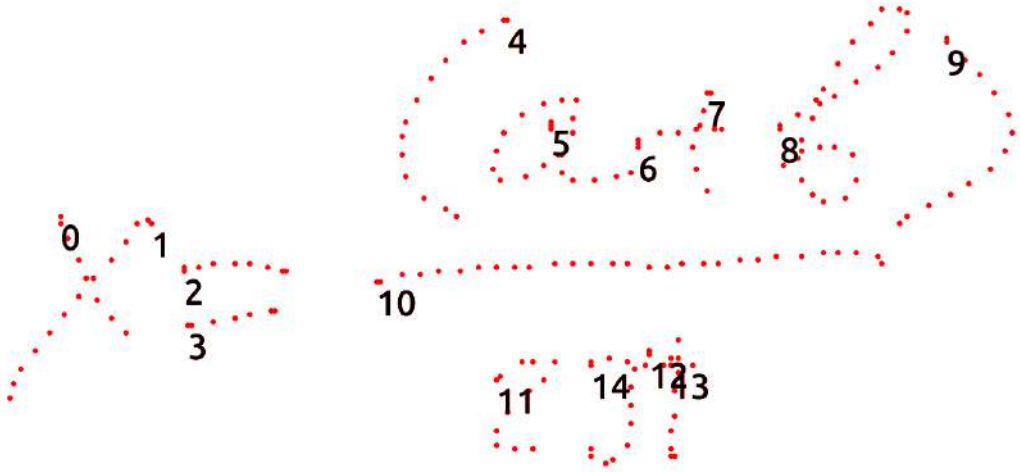
Рис. 3.5. Етапи попередньої обробки штрихів

З огляду на відсутність єдиної базової лінії, першою відмінністю є те, що нормалізація проводиться, виходячи з висоти медіанного символу, а не розміру висоти текстового рядка відносно базової лінії. У розрахунку висоти медіанного символу не враховуються символи, розмір яких сильно витягнутий по одній із координат. Найчастіше витягнуту геометрію мають штрихи, які відносяться до «подовжуваних» символів або дуже маленьких символів, таких як кома.

Другою важливою особливістю є алгоритм для переупорядковування штрихів, який заснований на попередньому аналізі структури виразу. Бажаний порядок жестів задається правилом зверху вниз для вертикальних елементів і зліва направо – для горизонтальних. Переупорядковування побудовано на рекурсивній кластеризації вхідних жестів на компоненти, де всередині кожного компонента має дотримуватися заданий порядок жестів. Для більш правильного визначення вертикального розташування елементів використовується структурний аналіз вхідних штрихів за допомогою пошуку спеціальних символів, таких як знак дробу та кореня. Таким чином, кластеризація проводиться з урахуванням дистанції між символами та результату визначення символів, які задають вертикальне розташування елементів. Результатом сегментації є дерево компонент, яке задає попередню структуру виразу. Усередині кожної компоненти проводиться упорядкування штрихів зліва направо. Потрібний порядок штрихів для всього МВ будується шляхом конкатенації штрихів з усіх компонент зліва направо та зверху вниз.



(а) Вхідні точки. Порядок штрихів: $x = \text{frac } a + b() + 2\pi$



(б) Результат попередньої обробки. Порядок штрихів: $x = (a + b)\text{frac } 2\pi$

Рис. 3.6. Приклад попередньої обробки точок для виразу $x = \frac{(a+b)}{2\pi}$

Рисунок 3.6 демонструє приклад попередньої обробки штрихів, де числами позначені порядкові номери жестів. Як бачимо, вхідні штрихи містять дуже велику кількість точок, та їх вхідний порядок відрізняється від бажаного: знак дробу був написаний перед чисельником і знаменником, символ + був написаний з «відстроченим» штрихом, дужки написані в самому кінці. В результаті попередньої обробки кількість точок була значно зменшена. Також порядок штрихів відповідає зазначеним вище вимогам: спочатку йдуть всі жести, які відносяться до символів чисельника, потім йде знак дробу, в кінці йдуть жести, які належать знаменнику. Усередині кожної горизонтальної групи порядок жестів строго відповідає правилу зліва направо.

3.4. Загальна архітектура нейронної мережі для сегментації та класифікації символів

Навчання проводиться за допомогою міні-пакетного градієнтного спуску та алгоритму оптимізації ADADELTA [188], який забезпечує надійну збіжність параметрів. Архітектура РНМ для розпізнавання та сегментації символів представлена на Рисунку 3.7. Структура НМ така: вхідний шар, два прямі та зворотні шари з шаром, який об'єднує їхні результати, Dense-шар та шар декодування НЧК. Загальна структура мережі складається з 9 шарів і загальної кількості ваг близько 300 000.

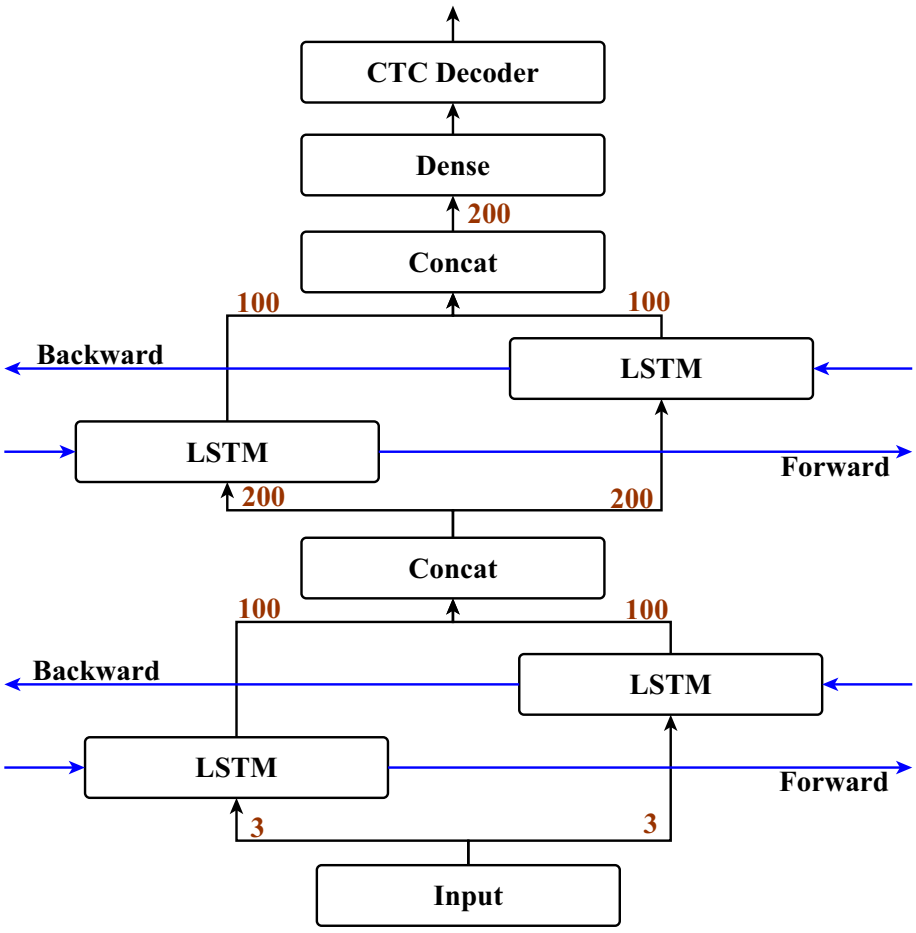
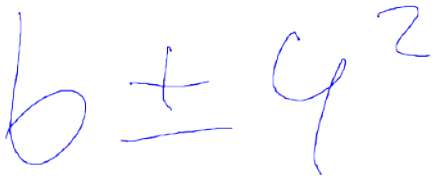


Рис. 3.7. Архітектура нейронної мережі для розпізнавання та сегментації символів

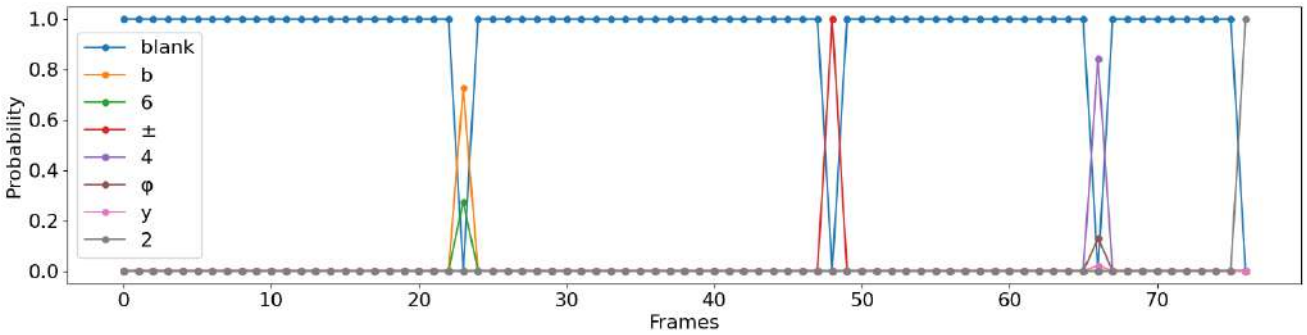
Вхідний шар отримує нормалізовані вектори ознак, де кожен вектор містить 3 ознаки: дельта x координати, дельта y координати та ознака натискання або відпускання пера (pen-up/pen-down). *Dropout* шар ($p = 0.5$) активний лише під час

навчання та призначений для підвищення продуктивності узагальнення. Умовні ймовірності класу символів отримуються на останньому шарі з функцією активації *softmax*. Під час декодування відбираються всі кандидати, крім *blank* символа, ймовірність яких більша певного порогового значення V_{dec} . Результатом декодування є вектор кандидатів сегментації символів $X = [\ (O_{start}, O_{end}, Z\ ]$. O_{start}, O_{end} визначають індекси початкової та кінцевої точок у вхідній послідовності, а $Z = [(L, V)])\ ]$ – вектор варіантів класифікації символів (L – мітка класу, V – її ймовірність).

На Рисунку 3.8 зображений приклад рукописного виразу та відповідний вихід рекурентної мережі з НЧК. На цьому прикладі видно чітку сегментацію, де в останній точці кожного символу відбувається сплеск ймовірностей символів, відмінних від *blank* символа. Для цього прикладу буде згенеровано результат, який містить 4 елементи $|X| = 4$. При цьому перший елемент має два варіанти класифікації $|X_0.Z| = 2$.



(а) Рукописний вираз $b \pm 4^2$



(б) Вихід рекурентної мережі з НЧК

Рис. 3.8. Приклад рукописного виразу $b \pm 4^2$ та відповідний вихід рекурентної мережі з НЧК

Застосування ДДКЧП для сегментації та класифікації елементів у 2D-послідовностях вводить вимоги до навчального набору даних. З досвіду бажа-

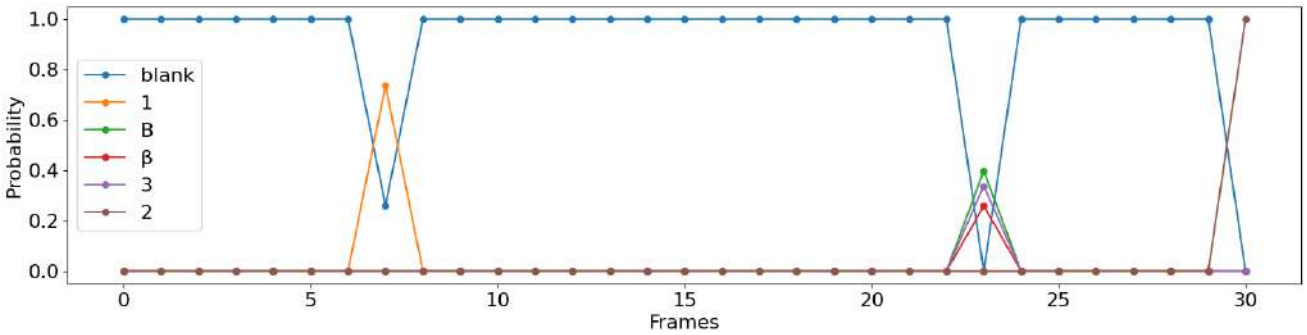
но, щоб послідовності з одного елементу були присутніми в навчальному наборі даних, оскільки без послідовностей з окремих елементів ДДКЧП не здатна виконувати завдання сегментації з високою ефективністю.

3.5. Класифікатор окремих символів

Запропонована архітектура РНМ здатна виконувати одночасно задачі сегментації та класифікації символів з високою ефективністю. Як було показано вище, в кінці кожного символу відбувається різка зміна ймовірностей, що свідчить про закінчення символу. Але при такому виході нейронної мережі інформація про початкову точку символу не представлена в явному вигляді. Єдиний спосіб визначення початкової точки є використання інформації про попередній сплеск. Але таке припущення не завжди вірне. Є випадки, коли початок символу не збігається з попереднім піком. Такий приклад зображений на Рисунку 3.9 для виразу 13^2. Так, перший пік визначає кінець символу ‘1’ з високою ймовірністю. Другий сплеск ймовірностей з приблизно однаковою ймовірністю передбачає символи ‘B’, ‘3’, ‘β’. Якщо спиратися тільки на інформації про поточний сплеск, то найбільш ймовірна послідовність символів буде ‘$1B2$’. При цьому інформація про геометрію символу ‘B’ буде невірною та відповідатиме розміру тільки другого штриха. Будемо називати результат такої сегментації невиразним або сумнівним.

Для зменшення таких помилок сегментації та підвищення точності класифікації символів у цій роботі пропонується інтегрувати додатковий ДДКЧП класифікатор символів. Головна ідея полягає в тому, щоб використовувати результати класифікації окремих символів більш легкою НМ для перевірки правильності сегментації головною НМ. Більше того, об’єднання результатів класифікації від двох мереж дає можливість поліпшити якість класифікації. На Рисунку 3.10 зображена архітектура НМ для класифікації окремих символів. Ця НМ також побудована на архітектурі ДДКЧП, але основною відмінністю від головної мережі є те, що вона має меншу кількість ваг ($\approx 50\,000$) і вирішує тільки завдання класифікації послідовності, а не маркування.

(а) Вхідні дані для виразу 13^2



(б) Вихід рекурентної мережі з невизначеним початком символу B

Рис. 3.9. Приклад рукописного виразу 13^2 з невиразною сегментацією

Уточнення результатів сегментації, яка отримана за допомогою головної НМ, проводиться шляхом перевірки перетину результатів класифікації від двох НМ. Для роботи з результатами двох класифікаторів ми будемо використовувати нечіткі множини та операції над ними, де отримана ймовірність визначає ступінь належності до множини.

Для кожної сумнівної сегментації викликається додатковий класифікатор окремих символів M, який повертає набір міток та їх ймовірності $Z = [(L, V)])$]. Далі будується набір I, який містить результат перетину двох класифікаторів. Якщо цей набір I не порожній та сума ймовірностей по всіх елементах більша порогового значення V_{merge}, то це означає неправильну сегментацію (помилковий пік). У цьому випадку приймається рішення про видалення символів, пов'язаних із помилковими піками, та створення одного символу, де результат класифікації заснований на перетині результатів двох окремих класифікаторів. У всіх інших випадках виконується об'єднання результатів двох класифікаторів із функцією належності $\mu_{A+B}(x) = \frac{\mu_A(x)+\mu_B(x)}{2}$. Алгоритм 3.1 містить псевдокод алгоритму вери-

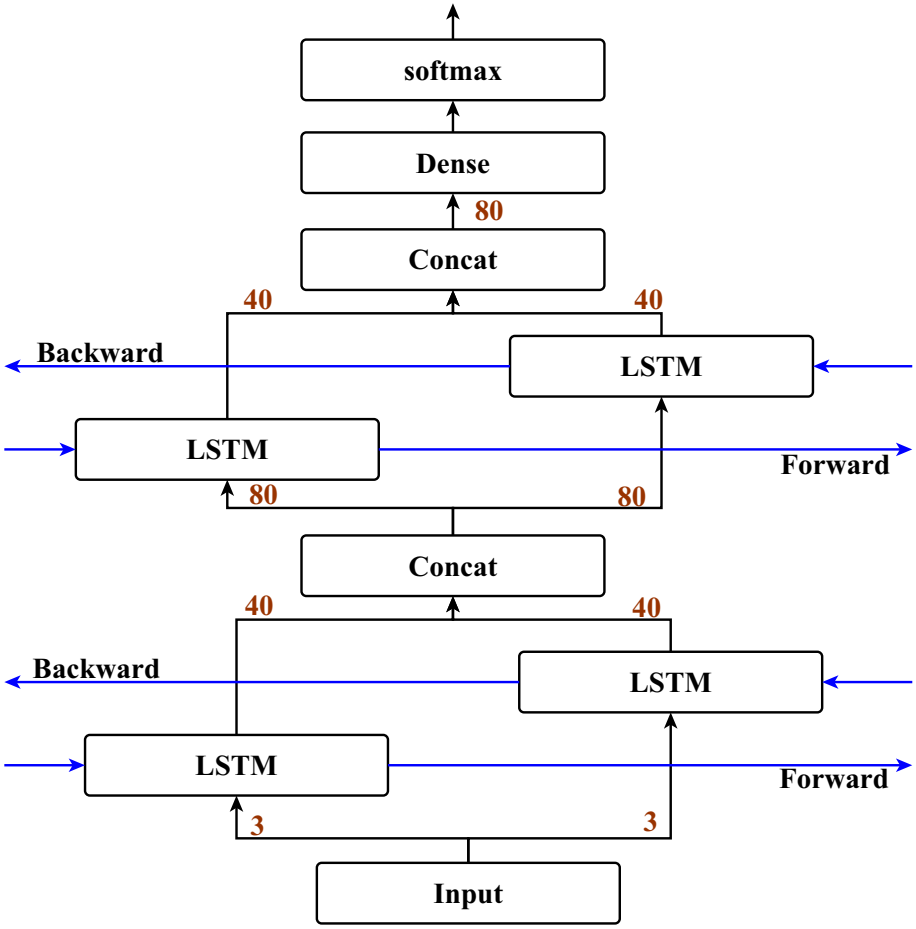


Рис. 3.10. Архітектура додаткової нейронної мережі для класифікації символів

фікації сегментації та об'єднання результатів класифікації головною і додатковою НМ.

3.6. Висновки до Розділу 3

Розділ містить стислий опис архітектури НМ, що має назву двонаправлена довга короткочасна пам'ять, та опис нейромережевої часової класифікації. Дані методи часто використовуються в задачах маркування послідовностей для 1D мов. Запропонована архітектура НМ вирішує одночасно два завдання: сегментації та класифікації символів.

Були розглянуті аспекти попередньої обробки штрихів з урахуванням особливостей написання математичних виразів, таких як наявність «подовжуваних» символів та велика варіативність розмірів символів. Одним з основних етапів по-

Алгоритм 3.1: Псевдокод алгоритму верифікації сегментації та об'єднання результатів класифікації головною і додатковою НМ

Algorithm `CharacterVerification()`

Input: A list of $N = |X|$ character segmentation and classification primitives $X = [\ (O_{start}, O_{end}, Z = [(L, V)])\ ]$.
Input: A single character classifier M.
Output: The modified list X.

for $i = 1$ **to** N **do**
  $p = 0$; `/* previous suspicious segmentation point */`
  $m =$ false ; `/* merge status */`
  `/* Search the previous suspicious segmentation point */`
  for $j = i - 1;\ j > 1$ **and** $X_j.Z_0.V < V_{sus}; j = j - 1$ **do**
    $p = j$;
  if $p \neq 0$ **then** `/* suspicious segmentation was found */`
    `/* Classify the concatenated points sequence */`
    $Z = M$.Classify$(X_p.O_{start}, X_i.O_{end})$;
    $I = X_i.Z \cap Z$; `/* intersection result from 2 models */`
    $v = \sum_{z \in I} z.V$; `/* sum of probabilities after merge */`
    if $v > V_{merge}$ **then** `/* is merge reliable? */`
      `/* Merge current symbol and the previous symbols with suspicious segmentation into one symbol */`
      $X_i.Z = I$;
      $X_i.O_{start} = X_p.O_{start}$;
      for $j = p$ **to** $i - 1$ **do**
        $D = D \cup X_j$; `/* mark` X_j `to remove */`
      $m =$ true;
  if $\neg m$ **then**
    `/* Union result from 2 models without merge */`
    $Z = M$.Classify$(X_i.O_{start}, X_i.O_{end})$;
    $X_i.Z = X_i.Z + Z$;
  NormalizeProbabilities$(X_i.Z)$; `/* ensure that` $\sum_{z \in X_i.Z} z.V = 1$ `*/`
`/* Remove merged elements */`
$X = X \setminus D$;
return X;

передньої обробки є переупорядковування штрихів, оскільки порядок написання МВ не визначений і залежить від уподобань користувача. Для забезпечення правильного порядку вхідних штрихів виконується попередній аналіз структури виразу за рахунок сегментації вхідних штрихів. При цьому правильна сегментація забезпечується не тільки геометричними особливостями штрихів, але й раннім визначенням спеціальних символів, таких як знак дробу. Запропонований метод дає можливість отримати більш стабільний та інваріантний опис РМВ з точки зору елементарних послідовностей штрихів, а отже, корисний як етап попередньої обробки для наступного кроку розпізнавання символів на основі РНМ.

На відміну від 1D-мов, де важлива тільки правильна послідовність елементів, для МВ якість сегментації символів є критично важливою для подальших етапів, таких як визначення просторових відношень між символами та виразами. Основним недоліком НЧК є те, що часто неможливо точно визначити межі символу у вхідній послідовності. Тому основним внеском у цьому розділі дисертаційної роботи є розробка додаткової НМ на основі архітектури ДДКЧП та алгоритму інтеграції, що забезпечує значне підвищення якості сегментації та класифікації. При цьому додаткова НМ є досить легкою (приблизно в 6 разів меншою за основну НМ). Ефективність додаткової НМ буде розглянуто в Розділі 7.

РОЗДІЛ 4

КЛАСИФІКАЦІЯ ПРОСТОРОВИХ ВІДНОШЕНЬ

Проблема класифікації просторових відношень полягає у визначенні математичних відношень між елементами математичного виразу [81]. Стислий огляд запропонованих методів наведено у Розділі 2.3. Потрібно зауважити, що останнім часом проблема класифікації просторових відношень у літературі майже не розглядалась через поширення наскрізних рішень. Також під час розгляду різних методів класифікації просторових ознак зазвичай автори вивчають проблему тільки з точки зору точності та не приділяють належної уваги швидкісним характеристикам.

У цьому розділі розглянута проблема класифікації просторових відношень, запропоновані геометричні ознаки, надано порівняння декількох методів класифікації з точки зору точності та швидкості. Аналіз здійснено на відкритих наборах даних CROHME 2016 та 2019.

4.1. Проблема класифікації просторових відношень

Припустимо, що ми коректно сегментували і класифікували символи на попередніх етапах. Тоді наступний етап структурного аналізу полягає в побудові найбільш ймовірного дерева математичного виразу на основі інформації про мітки класів символів, розміри та розташування цих символів. Однак математична нотація має двовимірну структуру, де різні просторові відношення частіше за все визначають операції або аргументи. Тобто перед побудовою дерева математичного виразу потрібно мати механізм класифікації просторових відношень між символами або підвиразами.

Як уже згадувалося, зазвичай виділяють 6 основних типів просторових відношень: ‘*Right*’ (AB), ‘*Subscript*’ (A_B), ‘*Superscript*’ (A^B), ‘*Under*’ ($\underset{B}{A}$), ‘*Over*’

($\overset{B}{A}$), ‘*Inside*’ ($\sqrt{B}$). Також список може включати ще 4 додаткові відношення: ‘*RootIndex*’ ($\sqrt[B]{}$), ‘*Left*’ (BA), ‘*PreSuperscript*’ (${}^{B}A$), ‘*PreSubscript*‘ (${}_{B}A$) (див. Рисунок 4.1). У цій роботі використовуються всі 10 типів відношень.

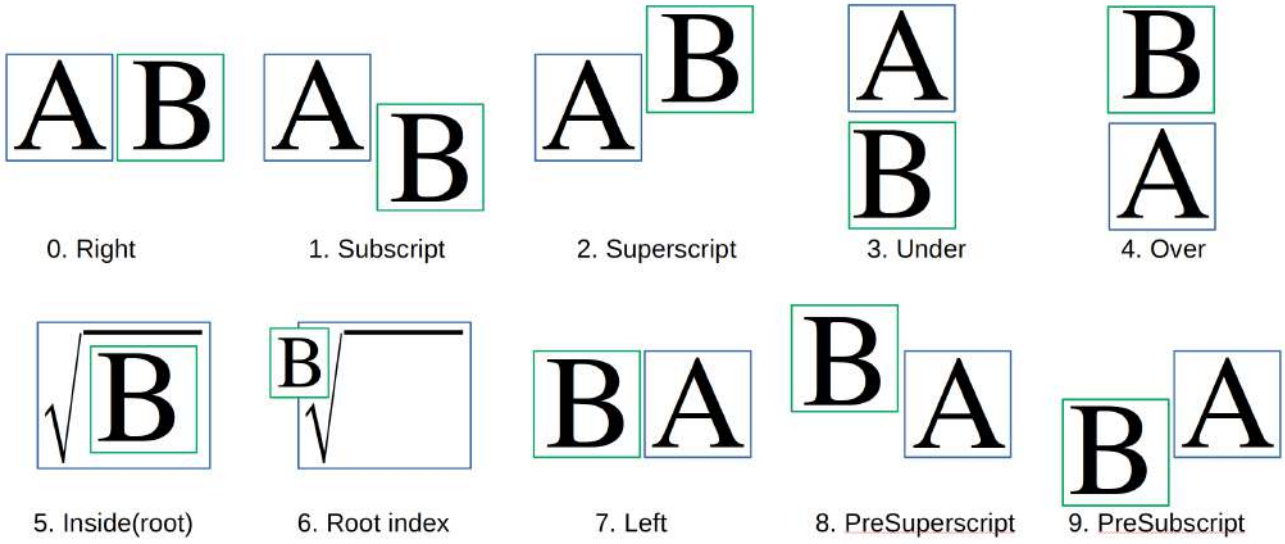
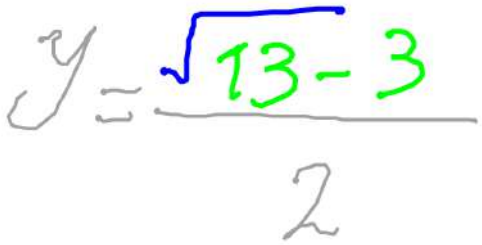


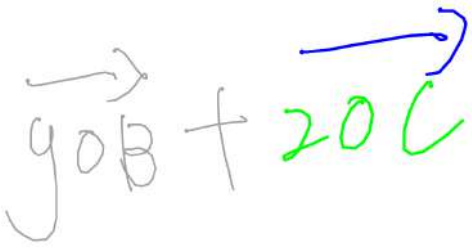
Рис. 4.1. Класи просторових відношень

Попри те, що кількість просторових відношень невелика, їх класифікація може бути дуже неоднозначною навіть для людини. Найчастіше проблема класифікації виникає саме у розпізнаванні рукописних МВ, де на класифікацію просторових відношень можуть суттєво впливати особливості почерку. Рисунок 4.2 демонструє приклади найбільш поширених двозначностей у РМВ.

Наведемо семантичне значення для основних типів відношень. ‘*Right*’ використовується для послідовного з’єднання чисел ‘58’, імен функцій ‘sin’ і змінних ‘$offset$’, з’єднання підвиразів через інші оператори ‘$a+b$’, а також часто застосовується як неявний оператор множення ‘ab’. ‘*Subscript*’ позначає індекс ‘A_i’ або може використовуватися як частина імені ‘T_{off}’. Як індекс може виступати будь-який підвираз. Найчастіше ‘*Superscript*’ позначає операції ступеня ‘x^2’, але також зустрічаються інші приклади, такі як позначення похідної ‘f'’ або градусів ‘$90°$’. Великі оператори зазвичай мають відношення ‘*Subscript/Superscript*’ ‘$\sum_i^n d_i$’ або ‘*Under/Over*’ ‘$\sum\limits_{i}^{n} d_i$’ для позначення параметрів (діапазону), де кожен параметр також є самостійним підвиразом. Наведені приклади є семантично однаковими, але відрізняються з точки зору граматики, тому ми не можемо ігнорувати ці від-

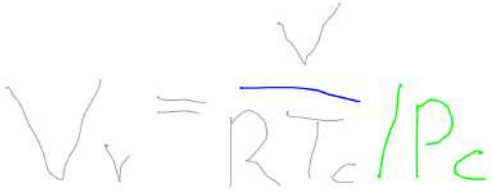


(а) $\sqrt{13} - 3$ або $\sqrt{13 - 3}$



(б) $\overrightarrow{2OC}$ або $\overrightarrow{OC}$

(в) S_n або Sn



(г) $\frac{V}{RT_c/P_c}$ або $\frac{V}{RT_c}/P_c$

Рис. 4.2. Приклади математичних виразів з неоднозначними відношеннями

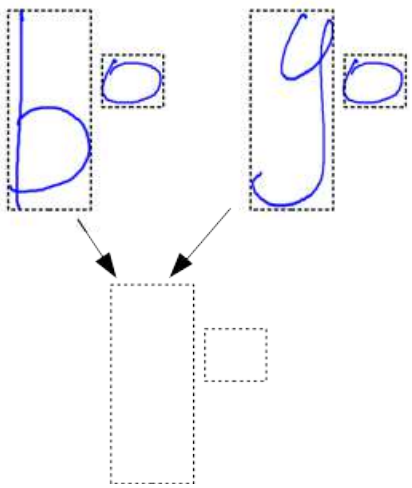
мінності у класифікації просторових відношень. Цей опис дуже стислий і далеко не повною мірою подає варіанти застосування просторових відношень у математичній нотації. У багатьох наукових та інженерних дисциплінах просторові відношення можуть використовуватися для різних цілей. Так, для позначення системи числення може застосовуватися ‘*Subscript*’ (101101_2). У хімії для опису молекулярної структури використовується хімічний знак елемента з індексом внизу, що вказує число атомів у молекулі ‘P_2O_3’.

Наведені приклади демонструють, що в математичній нотації майже не існує обмежень у конструюванні просторових зв’язків. Також мови запису математичних нотацій, такі як MathML або LaTeX, надають механізми визначення просторових зв’язків, але жодним чином не контролюють семантики виразу. Тому у класифікації просторових зв’язків дуже складно спиратися на семантичне значення розпізнаних символів.

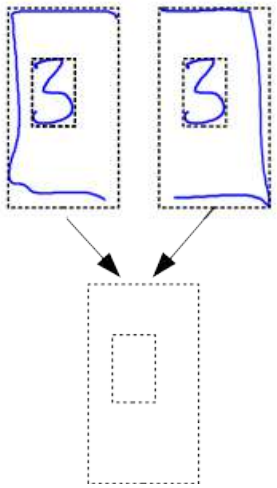
4.2. Типи символів та «body box»

Ця робота заснована на геометричних ознаках символів. Але підготовка набору ознак на основі обмежувального прямокутника («bounding box» або «bounding

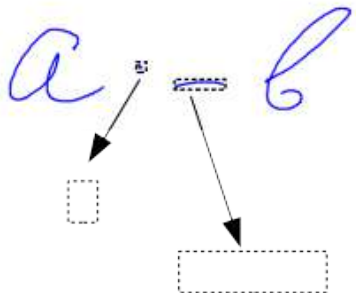
rectangle») не враховує особливостей символів, які впливають на класифікацію просторових зв'язків між елементами висловлювання. На Рисунках 4.3(а), 4.3(б) зображено приклади виразів, де обмежувальні прямокутники мають однакове розташування щодо один одного, але при цьому відповідають різним типам просторових відношень. На Рисунку 4.3(в) показано приклад обмежувальних прямокутників, якщо поруч стоять символи, у яких один із розмірів (ширина або висота) дуже маленький. Як видно з наведених прикладів, ознаки, які обчислюються тільки за допомогою обмежувальних прямокутників, негативно позначаться на якості класифікатора.



(а) Однакові прямокутники для '*Subscript*' та '*Superscript*' типів відношень



(б) Однакові прямокутники для '*Right*' та '*Left*' типів відношень



(в) Приклад обмежувальних прямокутників для виразу $a \cdot -b$

Рис. 4.3. Приклади неоднозначності у застосуванні обмежувальних прямокутників

У статті [97] автори запропонували концепцію «body box», яка враховує інформацію про геометричні особливості символів. Вони запроваджують 4 класи символів: *Ascendant*, *Descendant*, *Normal* та *Big*. «Body box» – це новий прямокутник B, який обчислюється шляхом зміни розмірів по Y-координаті. Узагальнена формула виводу може бути представлена у вигляді:

$$\begin{aligned} B_t &= R_t - K_1 * R_h \\ B_b &= R_b + K_2 * R_h \\ B_l &= R_l \\ B_r &= R_r, \end{aligned} \tag{4.1}$$

де R – координати обмежувального прямокутника; B – прямокутник «body box»; K_1 та K_2 – коефіцієнти, які визначаються класом символу. Кожен прямокутник має такі властивості: R_l – ліва координата прямокутника; R_r – права координата прямокутника; R_t – верхня координата прямокутника; R_b – нижня координата прямокутника; R_w – довжина прямокутника; R_h – ширина прямокутника; R_{cx} – центр прямокутника по координаті X; R_{cy} – центр прямокутника по координаті Y.

Але запропонований підхід орієнтований на зменшення конфліктів між відношеннями ‘*Right*’, ‘*Subscript*’ та ‘*Superscript*’ для звичайних символів. Для подолання обмежень, пов’язаних з неоднозначністю між просторовими відношеннями ‘*Right*’ та ‘*Left*’, в цій роботі пропонується збільшити кількість класів символів, при обчисленні «body box» враховувати сусідній символ та перетворювати розміри по X-координаті. Запропоновані класи символів зазначені на Рисунку 4.4.

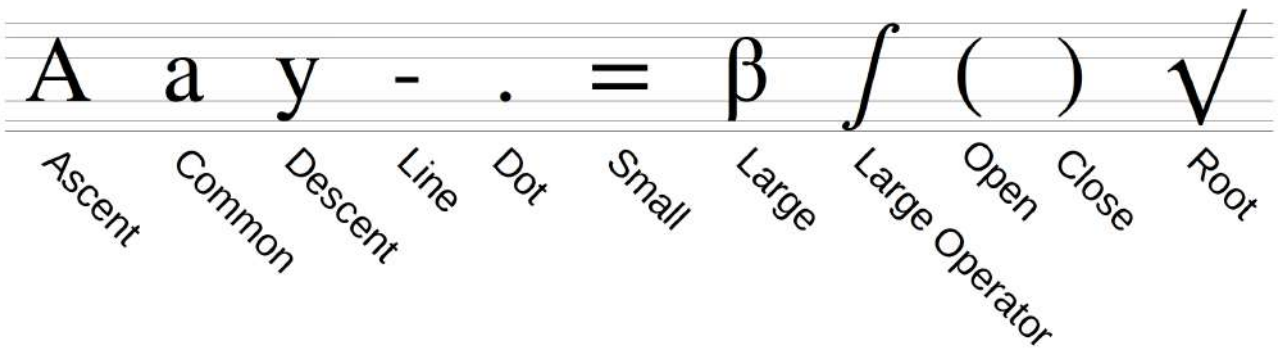


Рис. 4.4. Класи символів за геометричними ознаками

Класи символів, такі яі ‘*Open*’ та ‘*Close*’, залежать від пов’язаного з ними виразу. Так, їхній розмір може змінюватися, особливо якщо вираз у дужках містить складні вирази, такі як дроби і радикали. Клас символів ‘*Dot*’ та ‘*Line*’ потребують особливої уваги тому, що неточності у їх написанні особливо позначаються

на класифікації просторових відношень, а також можуть мати негативний вплив на процес навчання моделі класифікатора.

Обчислення «body box» буде здійснюватися за такою формулою:

$$\begin{aligned} B_t^A &= R_t^A + K_t^A * R_h^A + K_t^B * R_h^B \\ B_b^A &= R_b^A + K_b^A * R_h^A + K_b^B * R_h^B \\ B_l^A &= R_l^A + K_l^A * R_w^A + K_l^B * R_h^B \\ B_r^A &= R_r^A + K_r^A * R_w^A + K_r^B * R_h^B, \end{aligned} \tag{4.2}$$

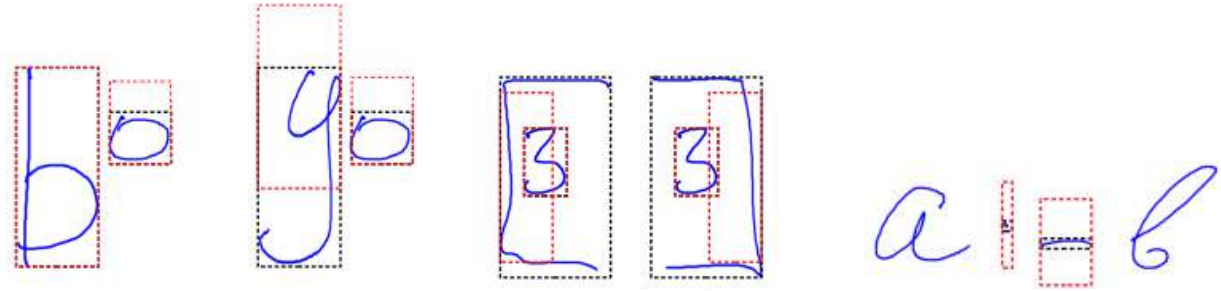
де R^A – координати обмежувального прямокутника для першого символу; R^B – координати обмежувального прямокутника для другого символу; B^A – прямокутник «body box» для першого символу; K^A та K^B – коефіцієнти, які визначаються класами сусідніх символів.

Якщо обидва суміжні символи належать до класу ‘*Dot*’ або ‘*Line*’, то обчислення «body box» визначається формулою:

$$\begin{aligned} B_t^A &= R_{cy}^A + K_t^A * R_h^* \\ B_b^A &= R_{cy}^A + K_b^A * R_h^* \\ B_l^A &= R_l^A \\ B_r^A &= R_r^A, \end{aligned} \tag{4.3}$$

де R^A – координати обмежувального прямокутника для першого символу; R^* – середній розмір символів у виразі; K^A – коефіцієнти, які визначаються класом символу A.

На Рисунку 4.5 показані приклади отриманих «body box». Такий підхід дає можливість враховувати геометричні властивості, знання про клас символу, а також локальний контекст (інформацію про сусідні символи). Це допомагає підготувати більш релевантний набір ознак для класифікатора.



(а) ‘*Ascendant*’ та ‘*Descendant*’ символи (б) Дужки (в) Вузькі елементи ‘*Line*’

Рис. 4.5. Порівняння ознак для класифікації просторових відношень на основі «body box» (червоний) та обмежувального прямокутника (чорний)

4.3. Набір ознак для класифікації просторових відношень

В цій роботі класифікація просторових відношень буде виконуватися на наборі з 10 ознак, які обчислюються за допомогою «body box» та обмежувального прямокутника. Ці ознаки мають багато спільного з ознаками, опублікованими в роботі [7]. Але при цьому є і суттєві відмінності, такі як застосування «body box» та обмежувального прямокутника одночасно та різні алгоритми нормалізації значень. Основні геометричні властивості для набору ознак зображені на Рисунку 4.6. Детальний опис набору ознак наведено у Таблиці 4.1.

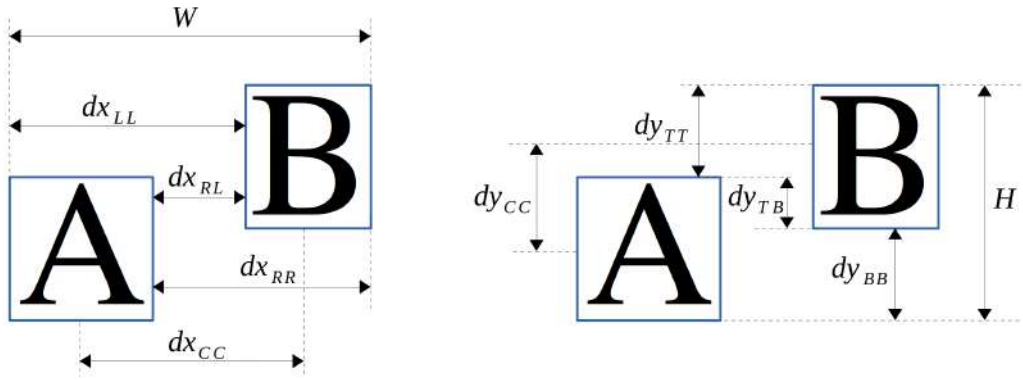


Рис. 4.6. Геометричні ознаки для класифікації просторових відношень

На відміну від попередніх робіт отриманий набір ознак не вимагає додаткової нормалізації для класифікації. Значення всіх ознак завжди знаходяться в діапазоні $[-1, 1]$. Очевидно, що деякі ознаки мають значну кореляцію. Наприклад, значення ознаки X_{CC} може бути отримано з ознак X_{LL}, X_{RR} та X_{RL}. Одночасне застосування ознак, які засновані на «body box» та обмежувальному прямокутни-

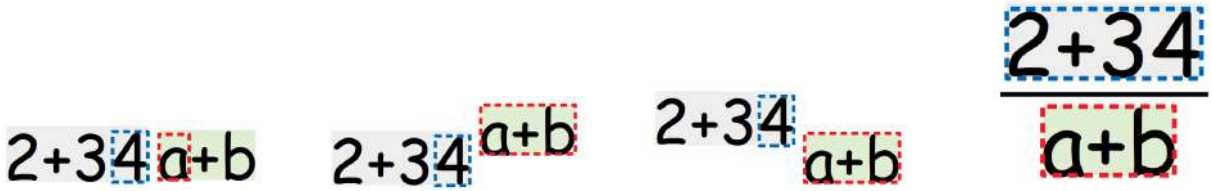
Таблиця 4.1. Геометричні ознаки для класифікації просторових відношень

Індекс	Позначення	Опис
0	$X_{RL} = \frac{dx_{RL}}{W}$	Відстань між правою координатою першого символу та лівою координатою другого символу («body box»)
1	$X_{LL} = \frac{dx_{LL}}{W}$	Відстань між лівими координатами символів («body box»)
2	$X_{RR} = \frac{dx_{RR}}{W}$	Відстань між правими координатами символів («body box»)
3	$Y_{TB} = \frac{dy_{TB}}{H}$	Відстань між верхньою координатою першого символу та нижньою координатою другого символу («body box»)
4	$Y_{TT} = \frac{dy_{TT}}{H}$	Відстань між нижніми координатами символів («body box»)
5	$Y_{BB} = \frac{dy_{BB}}{H}$	Відстань між верхніми координатами символів («body box»)
6	$X_{CC} = \frac{dx_{CC}}{W}$	Відстань між центрами вздовж координати X («body box»)
7	$Y_{CC} = \frac{dy_{CC}}{W}$	Відстань між центрами вздовж координати Y («body box»)
8	$Y^*_{TT} = \frac{dy^*_{TT}}{H^*}$	Відстань між нижніми координатами символів (обмежувальний прямокутник)
9	$Y^*_{BB} = \frac{dy^*_{TT}}{H^*}$	Відстань між верхніми координатами символів (обмежувальний прямокутник)

ку, зумовлене різноманітністю почерків. Коефіцієнти для розрахунку «body box» є усередненими для всіх почерків і в деяких випадках можуть негативно позначатися на аналізі просторового положення елементів. Особливо таке розмаїття почерків позначається в аналізі відношень для ‘*Descent*’ символів, таких як ‘y’, ‘μ’, ‘g’, ‘β’ та інші. У наступному підрозділі порівняємо різні методи класифікації та зробимо аналіз впливу додаткових ознак на якість класифікації і швидкість роботи.

Також важливою особливістю є визначення елементів, між якими проводиться класифікація відношень. У РМВ відсутня чітка базова лінія. Тобто кожен

наступний символ вносить додаткове відхилення. Також часто увесь вираз може бути написаний під невеликим кутом. Для того, щоб мінімізувати помилку, пропонується обчислювати ознаки для різних елементів підвиразів. При цьому самі елементи визначаються ймовірним типом відношень. На Рисунку 4.7 зображені приклади таких залежностей. Так, для перевірки відношення ‘*Right*’ набір ознак розраховується, враховуючи тільки крайні елементи підвиразів. У побудові ‘*SuperScript*’/‘*SubScript*’ операцій застосовується крайній правий елемент з лівого підвиразу та правий підвираз. Для всіх інших випадків ознаки розраховуються на основі усіх символів у підвиразі.



(а) ‘*Right*’ (б) ‘*SuperScript*’ (в) ‘*SubScript*’ (г) ‘*Under*’/‘*Over*’

Рис. 4.7. Приклади елементів, для яких обчислюються ознаки залежно від типу ймовірного просторового відношення

4.4. Порівняння методів класифікації просторових відношень

Для перевірки результатів класифікації просторових відношень був сформований навчальний набір на закритих даних. Тестування здійснювалося на двох відкритих наборах даних CROHME 2016 [121] та CROHME 2019 [112]. Зазвичай у літературі проводиться порівняння методів на кожному наборі окремо.

В результаті обробки було отримано 854686 відношень для навчального набору даних. Тестові набори містили 25008 і 25250 відношень відповідно. Рисунок 4.8 показує кількість зразків відношень для кожного класу в навчальному наборі. Рисунок 4.9 відповідно відображає кількість зразків у тестовому наборі. В обох наборах даних відношення ‘*Right*’ і ‘*Left*’ значно переважають решту типів відношень. Також обидва набори даних містять приблизно однакову пропорцію зразків на кожен клас.

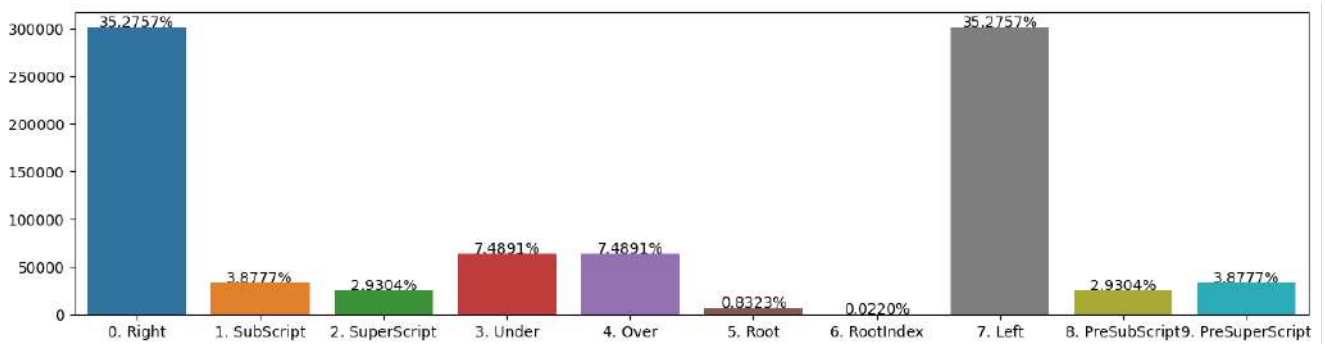


Рис. 4.8. Кількість зразків просторових відношень у навчальному наборі

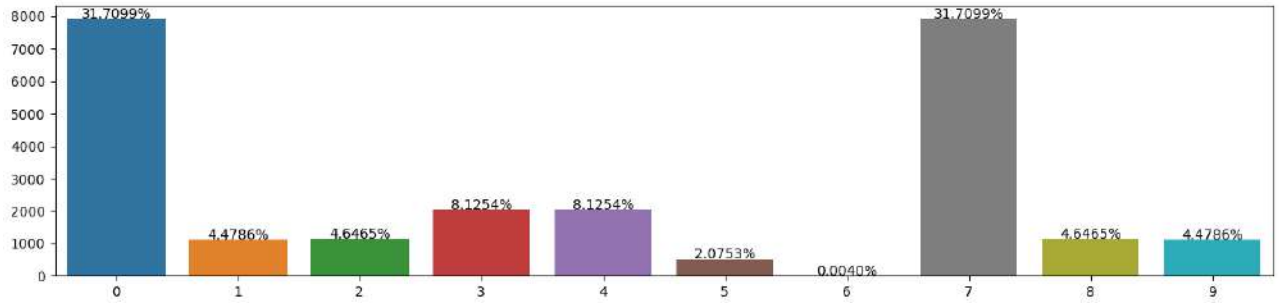


Рис. 4.9. Кількість зразків просторових відношень у тестовому наборі CROHME 2019

Також для оцінки кожної ознаки наведемо розподіл значень (Рисунок 4.10) та гістограму значень (Рисунок 4.11). На підставі плавної форми багатьох графіків можна говорити про те, що багато класів мають між собою великі колізії і відсутні чіткі межі між ними. Це добре видно в аналізі ознак Y_{TT} (feature_4) та Y_{BB}(feature_5), які більшою мірою відповідають за класифікацію відношень ‘*Right*’, ‘*Subscript*’ та ‘*Superscript*’. Також можна простежити позитивний ефект від застосування «body box» замість обмежувального прямокутника при порівняльному аналізі ознак Y_{TT} (feature_4) та Y^*_{TT}(feature_8).

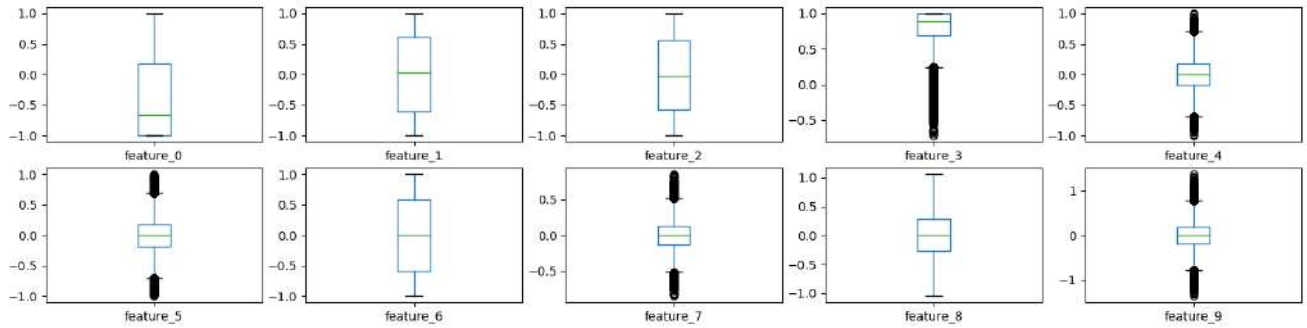


Рис. 4.10. Розподіл значень для кожної ознаки просторового відношення у навчальному наборі

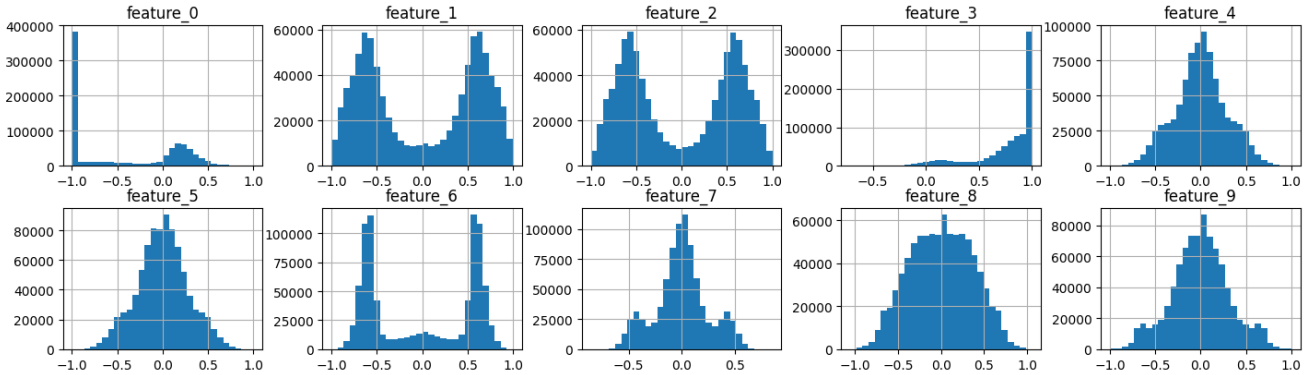


Рис. 4.11. Гістограма для кожної ознаки просторового відношення у навчальному наборі

Для вибору алгоритму класифікації було виконано порівняння найбільш поширених із них. У порівнянні оцінювалися метрики якості та швидкодії. Таблиці 4.2 та 4.3 містять порівняльні показники алгоритмів, які продемонстрували найкращі результати. Результати інших алгоритмів не наводяться через значно нижчий показник якості. Як бачимо, найкращий показник якості має багатошаровий перцептрон (Multi-layer Perceptron). В експерименті ми використовували логістичну функцію активації (sigmoid), один прихований шар та нормовану експоненційну функцію (softmax) у вихідному шарі.

З точки зору швидкодії найкращий результат отриманий при застосуванні дерева ухвалення рішень (Decision Tree). Для уникнення перенавчання максимальна глибина дерева була обмежена до 11 рівнів, а кількість прикладів для поділу не могла бути менше ніж 200. При відставанні на 0.1%–0.2% в якості швидкість роботи такого алгоритму більша ніж у 6 разів. Незважаючи на явну перевагу в швидкодії, підхід на основі дерева прийняття рішень вимагає більшого обсягу пам'яті. Слід також зазначити, що обчислювальна складність обох алгоритмів становить $O(1)$, але обчислювальні затрати на одну операцію у багатошарового перцептрона значно вищі.

Як уже згадувалося, деякі ознаки мають кореляцію. Для перевірки впливу цих ознак на якість і швидкість було проведено порівняння із застосуванням двох алгоритмів, які показали найкращі результати. Перевірка здійснювалася за допомогою двох додаткових наборів ознак. Перший набір містить тільки 6 ознак з ви-

Таблиця 4.2. Порівняння різних класифікаторів для визначення просторових відношень на CROHME 2016

Класифікатор	Precision	Recall	F1-score	Time (sec)
Logistic Regression	0.95546	0.95549	0.95543	0.00921702
Decision Tree	0.96082	0.96069	0.96062	0.00324559
Random Forest (3 trees)	0.95637	0.95633	0.95544	0.01214194
K-NN classifier	0.94301	0.93518	0.93763	**0.00290203**
Gaussian Naive Bayes	0.94327	0.93414	0.93669	0.00958561
Multi-layer Perceptron (1×10)	**0.96222**	**0.96225**	**0.96221**	0.02120852
Support Vector Machine	0.95717	0.95745	0.95723	121.201831

Таблиця 4.3. Порівняння різних класифікаторів для визначення просторових відношень на CROHME 2019

Класифікатор	Precision	Recall	F1-score	Time
Logistic Regression	0.95792	0.95719	0.95746	0.01212334
Decision Tree	0.96585	0.96491	0.96528	**0.00311660**
Random Forest (3 trees)	0.96602	0.96511	0.96543	0.01192522
K-NN classifier	0.94834	0.93782	0.94107	0.00320339
Gaussian Naive Bayes	0.94910	0.93279	0.93760	0.00974774
Multi-layer Perceptron (1×10)	**0.96717**	**0.96602**	**0.96648**	0.02266788
Support Vector Machine	0.96015	0.95941	0.95970	122.222898

користанням «body box» і не має кореляційних ознак. Другий набір включає дві ознаки, що мають кореляцію з попередніми 6 ознаками. Результати порівняння наведені в Таблиці 4.4. Додавання нових ознак дало можливість зменшити помилку розпізнавання на 15%, при цьому збільшення ознак практично не впливає на швидкодію.

На Рисуноку 4.12 наведена матриця невідповідностей. Найбільша помилка класифікації пов'язана з неправильною класифікацією '*Right*' як '*Subscript*', '*Over*' як '*Root index*', та '*Left*' як '*PreSubscript*'. Помилки неправильної класифікації були проаналізовані і приклади найбільш поширених типів помилок представлені на Рисунку 4.13.

Досить велика група помилок пов'язана з написанням під значним кутом (Рисунок 4.13(а), 4.13(б), 4.13(в)). Звичайно, що частина помилок буде виправлена на наступному етапі, але загальне рішення вимагає визначення кута нахилу виразу

Таблиця 4.4. Порівняння різних наборів ознак для визначення просторових відношень

Класифікатор та набір ознак	CROHME 2016		CROHME 2019	
	Accuracy	Time (sec)	Accuracy	Time (sec)
Decision Tree (1-6 feaures)	0.95166	0.00328922	0.95790	0.00328159
Decision Tree (1-8 feaures)	0.95182	0.00318121	0.95533	0.00318932
Decision Tree (1-10 feaures)	0.96069	0.00324559	0.96491	0.00311660
MLP (1-6 feaures)	0.95537	0.02057695	0.95857	0.02067494
MLP (1-8 feaures)	0.95705	0.02069139	0.96071	0.02073407
MLP (1-10 feaures)	0.96225	0.02120852	0.96602	0.02266788

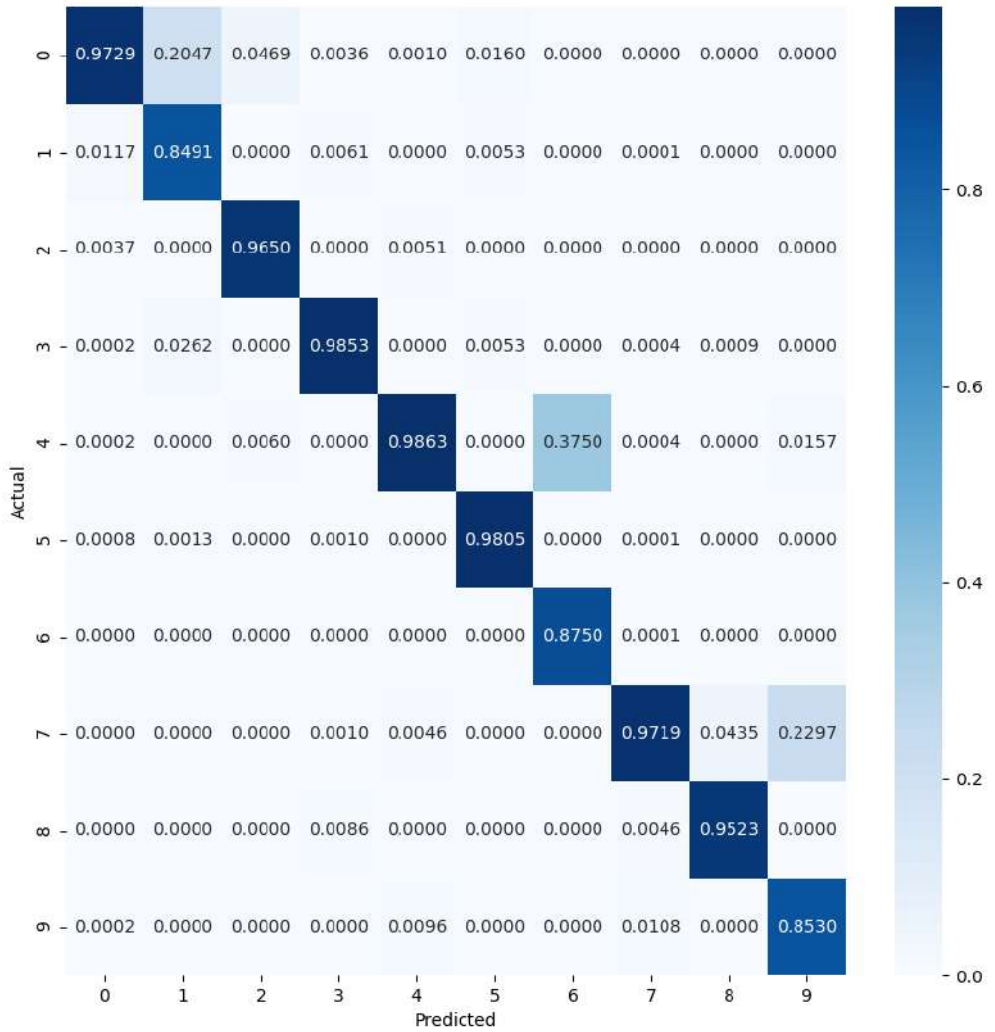
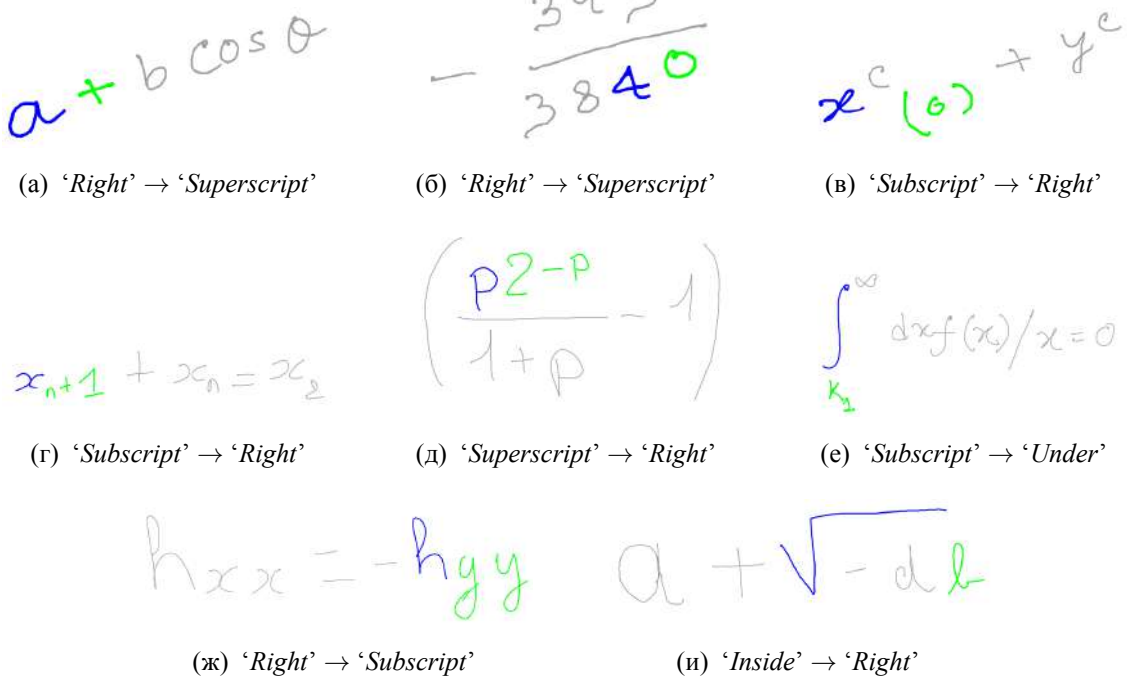


Рис. 4.12. Матриця невідповідностей класифікації просторових відношень (MLP, CROHME 2019)

і обертання всього виразу на етапі попередньої обробки. Однак помилки у визначенні кута нахилу можуть негативно позначитися на розпізнаванні символів і структури виразу. Наведемо приклади виразів, для яких дуже складно визначити



(а) ‘*Right*’ → ‘*Superscript*’ (б) ‘*Right*’ → ‘*Superscript*’ (в) ‘*Subscript*’ → ‘*Right*’

(г) ‘*Subscript*’ → ‘*Right*’ (д) ‘*Superscript*’ → ‘*Right*’ (е) ‘*Subscript*’ → ‘*Under*’

(ж) ‘*Right*’ → ‘*Subscript*’ (и) ‘*Inside*’ → ‘*Right*’

Рис. 4.13. Приклади помилок у класифікації просторових відношень

правильно кут нахилу: e^{2x^2+y}, $2^{2^{2^2}}$, A_{offset}. Зазвичай у таких виразах значна частина символів належить до ‘*Subscript*’ та ‘*Superscript*’.

Помилки пов’язані з неправильною класифікацією просторових відношень ‘*Subscript*’/’*Under*’ та ‘*Superscript*’/’*Over*’ навколо великих операторів 4.13(е), коли параметри оператора мають несумісні просторові відношення, зазвичай виправляються шляхом використання найбільш наближеного типу відношень. У цьому випадку не гарантується правильна структура, але забезпечується правильна семантика виразу. Помилки, пов’язані з просторовими відношеннями для ступеня кореня, можуть бути легко виправлені на наступному рівні під час побудови виразу. Також багато неточностей (приклад на Рисунку 4.13(б)) можуть бути компенсовані за рахунок застосування додаткових механізмів, таких як модель граматики і проста модель мови (ММ). Але часто не вдається повністю виправити помилки, оскільки запропоновані варіанти класифікації відношень призводять до допустимих варіантів виразу. Приклад такого виразу показаний на Рисунку 4.13(е), де $\sqrt{dl}$ і $\sqrt{dl}$ є цілком правильними. Крім представлених на зображенні прикладів, ще можна навести приклади, в яких рукописне зображення не відповідає основній правді (*ground truth*).

У багатьох випадках помилки можуть бути автоматично виправлені за рахунок контексту всього виразу. Але такий механізм може вимагати створення сотні або тисячі правил чи інтеграції громіздких НМ. Спосіб, що базується на НМ, зазвичай зустрічається в наскрізних рішеннях, у яких результат розпізнавання визначається за рахунок з'єднання двох нейронних мереж, де перша мережа відповідає за розпізнавання, а друга – за контекст математичного виразу і зазвичай називається модель мови. У такому разі розміри другої НМ можуть перевищувати розміри основної мережі.

4.5. Висновки до Розділу 4

У цьому розділі розглянута задача класифікації просторових відношень, яка є невід'ємним атрибутом у вирішенні задачі побудови структури в двовимірних мовах, таких як математична нотація. Для розв'язання цього завдання була покращена концепція «body box» за рахунок більшої деталізації класів символів та локального контексту, під яким розуміють інформацію про тип сусідніх символів та їхні геометричні властивості.

Для вирішення цього завдання також було запропоновано набір із 10 геометричних ознак. Ефективність такого набору була перевірена за допомогою кількох алгоритмів класифікації на відкритих тестових наборах даних. Кращі результати були отримані у застосуванні багатошарового перцептрону та дерева ухвалення рішень. Перевагою першого алгоритму є досить висока точність класифікації і незначний обсяг необхідної пам'яті. Водночас, основна перевага другого алгоритму – це швидкість роботи при незначному відставанні в точності (0.1%–0.2%) порівняно з першим. Також розділ містить короткий аналіз прикладів з неправильною класифікацією, розглянуто можливі варіанти вирішення цих проблем.

У подальшій роботі буде використано запропонований набір ознак у комбінації з деревом ухвалення рішень, оскільки саме ця комбінація більшою мірою відповідає вимогам щодо обмеження обчислювальних ресурсів.

РОЗДІЛ 5

СТРУКТУРНИЙ АНАЛІЗ МАТЕМАТИЧНОГО ВИРАЗУ

Завдання цього етапу – це побудова найбільш ймовірного представлення МВ. Основна складність розпізнавання МВ полягає в неоднозначностях, що виникають під час сегментації та класифікації символів та класифікації просторових відношень. Тобто вхідними даними для виконання структурного аналізу будуть результати класифікації кожної моделі з оцінкою ймовірностей. Також часто структурний аналіз називають алгоритмом розбору (*parsing algorithm*).

Як було зазначено в Розділі 2, підходи для структурного аналізу класифікують на графові і граматичні. Графові методи побудови математичного виразу базуються на геометричних ознаках штрихів та символів. Граматичні підходи спираються на формалізм граматик та демонструють кращу якість розпізнання, але мають значно більшу обчислювальну складність порівняно з графовими методами.

У цьому розділі наводиться визначення поняття двовимірної стохастичної контекстно-вільної граматики (2D-СКВГ), розглядаються правила специфікації граматики та метод розрахунку ймовірностей правил виводу (підрозділ 5.1). Далі міститься опис мовної моделі (підрозділ 5.2) та алгоритму розбору МВ (підрозділ 5.3), який базується на алгоритмі Кокке-Янгера-Касамі (КЯК) та забезпечує інтеграцію усіх моделей. У наступному підрозділі 5.4 розглядаються питання, пов’язані з внеском кожної моделі і навчанням вагових коефіцієнтів на основі генетичного алгоритму. Особлива увага приділяється оптимізації складності алгоритму (підрозділ 5.5) за рахунок застосування різних прийомів, які спираються на методи обчислювальної геометрії, теорії графів та динамічного програмування.

5.1. Двовимірна стохастична контекстно-вільна граматика

Контекстно-вільна граматика (КВГ) широко застосовується в інформатиці та є однією з 4-х типових граматик в ієрархії Чомскі [33]. Нею задається граматична структура більшості мов програмування. Потужний формалізм КВГ також здатний представити структуру природних мов. Розширення КВГ для двовимірних мов добре вивчені та поширені в задачах розпізнавання МВ [34, 123, 183].

Означення 5.1. Контекстно-вільна граматика G визначається як четвірка [73]:

$$G = (N, T, P, S) \tag{5.1}$$

де: $S \in N$; N та T скінченні множини, що не перетинаються; P скінченна підмножина $N \times (N \cup T)^*$.

Використовують такі назви: N – множина нетермінальних символів, T – множина термінальних символів, P – множина правил виводу, S – множина початкових символів.

Нормальна Форма Чомського (НФЧ) встановлюється для приведеної КВГ, де правила виводу мають таку форму:

$$\begin{aligned} A &\Longrightarrow BC, \quad or \\ A &\Longrightarrow \alpha, \quad or \\ S &\Longrightarrow \varepsilon, \end{aligned} \tag{5.2}$$

де: A, B та C – нетермінальні символи ($A, B, C \in N$); α – термінальний символ ($\alpha \in T$); ε позначає порожнє слово.

Означення 5.2. Синтаксичний аналіз (парсинг, parsing) – це процес аналізу вхідної послідовності символів з метою розбору граматичної структури згідно із заданою формальною граматикою.

Означення 5.3. Синтаксичний аналізатор (parser) – це програма, яка виконує синтаксичний аналіз.

Означення 5.4. Граматику називають багатозначною, якщо слово w мови $L(w \in L(G))$ в граматиці G може бути отримане кількома різними способами [102].

Означення 5.5. Стохастична контекстно-вільна граматика (СКВГ) визначається п'ятіркою:

$$G = (N, T, P, S, R) \tag{5.3}$$

де: R – це набір ймовірностей правил виводу $|R| = |P|$.

Означення 5.6. Двовимірна стохастична контекстно-вільна граматика (2D-СКВГ) – це узагальнення СКВГ, де термінальні і нетермінальні символи описують двовимірні області. Ця граматика у НФЧ дає два типи правил: термінальні та бінарні. Термінальні правила $A \Longrightarrow \alpha$ представляють математичні символи, які в кінцевому підсумку є термінальними символами 2D-СКВГ. Бінарні правила виводу $A \overset{r}{\Longrightarrow} BC$ мають додатковий параметр r, який представляє дане просторове відношення, і його інтерпретація полягає в тому, що області B і C мають бути просторово розташовані відповідно до відношення r.

У Розділі 4 було представлено типи просторових відношень і дано короткий аналіз їх семантичних значень. Для позначення типу просторових відношень у правилах виводу будемо використовувати таку нотацію: '*Right*' ($A \overset{\rightarrow}{\Longrightarrow} B$), '*Subscript*' ($A \overset{\searrow}{\Longrightarrow} B$), '*Superscript*' ($A \overset{\nearrow}{\Longrightarrow} B$), '*Under*' ($A \overset{\downarrow}{\Longrightarrow} B$), '*Over*' ($A \overset{\uparrow}{\Longrightarrow} B$), '*Inside*' ($A \overset{\square}{\Longrightarrow} B$), '*Left*' ($A \overset{\leftarrow}{\Longrightarrow} B$), '*PreSuperscript*' ($A \overset{\nwarrow}{\Longrightarrow} B$), '*PreSubscript*' ($A \overset{\swarrow}{\Longrightarrow} B$), '*RootIndex*' ($A \overset{\surd}{\Longrightarrow} B$).

Приклад правил виводу в нотації 2D-СКВГ:

$$\begin{aligned}
TERM &\Longrightarrow symbol \\
TERM &\Longrightarrow number \\
LEFT &\overset{\rightarrow}{\Longrightarrow} operator \quad TERM \\
EXPR &\overset{\rightarrow}{\Longrightarrow} TERM \quad LEFT \\
DENOM &\overset{\downarrow}{\Longrightarrow} hline \quad TERM \\
FRAC &\overset{\downarrow}{\Longrightarrow} TERM \quad DENOM
\end{aligned}$$

Бінарні правила виводу для відношень *Right* та *Left* відповідають за об'єднання елементів, які знаходяться на одній базовій лінії. Решта бінарних правил призначені для побудови підвиразів з двох елементів, що знаходяться на різних базових лініях.

5.1.1. Алфавіт математичних виразів та термінальні символи

Алфавіт математичних виразів містить понад 1,500 символів. У цій роботі розглядається тільки підмножина, яка складається з близько 200 символів і включає такі основні групи: цифри ($0-9$), латинські символи ($a-z, A-Z$), грецькі символи (α, β, γ тощо), унарні оператори ($+$, !, $\neg$ тощо), оператори відношення ($<$, $\gg$, $=$ тощо), бінарні оператори ($\times$, $\cap$, $\div$ тощо), великі оператори ($\sum$, $\int$, $\prod$ тощо), стрілки ($\rightarrow$, $\Leftrightarrow$, $\uparrow$ тощо), елементи теорії множин та математичної логіки ($\varnothing$, $\mathbb{N}$, $\in$, $\forall$ тощо), знаки та символи в геометрії ($\angle$, $\cong$, $\perp$ тощо), роздільники та дужки (|, [, } тощо), фінансові знаки (\$, £тощо). Також багато символів мають різні значення залежно від виразу та положення у виразі, для прикладу $\sum$ – сума або Σ – сігма.

Також граматичні правила можуть застосовуватися для корекції помилок розпізнавання, пов'язаних зі схожим написанням багатьох символів. Таким чином, правила термінальних символів засновані на списку підтримуваних символів, де кожному символу відповідає не менше ніж одне правило. Оскільки кількість правил величезна, то розділ не містить опис усіх правил, а наведено лише приклади таких правил.

$$
\begin{aligned}
SYMB\_DIGIT &\Longrightarrow 1 \\
SYMB\_DIGIT &\Longrightarrow | \\
SYMB\_BINARY &\Longrightarrow | \\
SYMB\_OPEN &\Longrightarrow | \\
SYMB\_CLOSE &\Longrightarrow | \\
SYMB\_DIGIT &\Longrightarrow \circ \\
SYMB\_BINARY &\Longrightarrow \circ \\
SYMB\_DEGREE &\Longrightarrow \circ
\end{aligned}
$$

5.1.2. Специфікація граматики

Незважаючи на те, що мова математичних виразів вважається формальною, але деякі вирази пишуться з неявними операторами, що може призвести до неправильного тлумачення. Прикладами таких виразів є: неявне множення $2ab$, $48 \div 2(3+2)$; неявні дужки $sin\frac{\pi}{2} + \frac{\pi}{3}$ та $\int 5x^{-4} + x^3 dx$. Наявність неявних операторів має бути врахована у побудові правил виду $A \Longrightarrow BC$. Але підтримка неявних операторів передбачає створення багатозначною граматики. Це може спричинити збільшення кількості правил виводу та складності алгоритму. Для зменшення багатозначності ця робота буде спиратися на правосторонню стратегію виводу.

Означення 5.7. Вивід слова w в граматиці G називається правостороннім, якщо в ньому на кожному кроці правило виводу застосовується до крайнього правого нетермінального символу в ланцюжку $A \to aA$.

Приклад застосування правосторонньої стратегії для побудови правил виводу:

$$NAME \overset{\rightarrow}{\Longrightarrow} LETTER \ \ LETTER$$
$$NAME \overset{\rightarrow}{\Longrightarrow} LETTER \ \ NAME$$

Побудова правил граматики в автоматичному режимі є найбільш зручним механізмом. Однак із праці [59] відомо, що неможливо побудувати КВГ лише на позитивних прикладах без посилання на статистичний розподіл:

- однак можна автоматично побудувати КВГ, якщо є як позитивні, так і негативні приклади;
- якщо відомі достатні додаткові попередні знання, такі як статистичний розподіл, можна вивчити КВГ лише з позитивними даними.

Для МВ відсутні набори з негативними прикладами. Методи побудови граматики із залученням тільки позитивних прикладів вивчалися в багатьох статтях [126, 142], та вони вимагають значної кількості прикладів. Існуючі загальнодоступні набори МВ побудовані на основі Вікіпедії [174]. Такі набори даних містять значну кількість прикладів з помилками, інформація про галузь застосування МВ відсутня або вимагає аналізу всього тексту статті. Також багато прикладів не потрапили в набір МВ, бо мають просту структуру, яку забезпечує 1D-текст, та не були позначені спеціальними тегами. Ще одним недоліком корпусу, побу-

дованого на базі Вікіпедії, є більш науковий склад МВ, при якому прості вирази представлені в невеликій кількості. Тому в цій роботі не застосовувалися автоматичні методи побудови КВГ, а побудову правил граматики було здійснено вручну.

5.1.3. Ймовірності правил виводу

Побудова ймовірностей правил виводу заснована на алгоритмі Вітербі [167] для пошуку найбільш ймовірної послідовності подій, що відбулися. Шлях Вітербі — це найбільш правдоподібна (найбільш ймовірна) послідовність прихованих станів. Нехай на виході спостерігається послідовність $y_1, y_2, \ldots, y_T$ подій. Тоді найбільш ймовірна послідовність станів $x_1, x_2, \ldots, x_K$ для спостережуваної послідовності може бути визначена за допомогою таких рекурентних співвідношень:

$$\begin{aligned} V_{1,k} &= P(y_1 \mid k) \cdot a_k \\ V_{t,k} &= \max_x(P(y_t \mid k) \cdot a_{x,k} \cdot V_{t-1,x}) \end{aligned} \tag{5.4}$$

де: $V_{t,k}$ – ймовірність найбільш вірогідного шляху, що закінчується в стані k в момент часу t; a_k – початкова ймовірність знаходження в стані i; $a_{x,k}$ – ймовірність переходу із стану x в стан k.

Шлях Вітербі може бути знайдений, запам'ятовуючи найбільш ймовірну послідовність станів x для кожного моменту часу, і представлений за допомогою виразу:

$$x_T = \arg\max_x(V_{T,x}) \tag{5.5}$$

Маючи справу з ймовірностями, великі операції множення, які ми виконуємо, можуть призвести до спустошення (*underflow*). Цього можна уникнути, якщо працювати з логарифмічними показниками. Тоді оцінка найбільш ймовірного шляху буде мати вигляд:

$$\begin{aligned} V_{1,k} &= \log(P(y_1 \mid k)) + \log(a_k) \\ V_{t,k} &= \max_x(\log(P(y_t \mid k)) + \log(a_{x,k}) + V_{t-1,x}) \end{aligned} \tag{5.6}$$

З точки зору граматики шлях Вітербі – це найбільш ймовірний МВ, де ймовірність правил виводу відповідає ймовірності переходу між станами. Розрахунок ймовірностей правила виводу здійснюється на тренувальному наборі МВ. Так для кожного виразу цього набору будується дерево розбору. Для кожного вузла цього дерева знаходиться відповідне правило виводу граматики за допомогою алгоритму розбору. Тоді ймовірність правила виводу $A \to \alpha$ визначається формулою:

$$p(A \to \alpha) = \begin{cases} \varepsilon, & \text{if } c(A \to \alpha) = 0 \\ \dfrac{c(A \to \alpha)}{c(A)}, & \text{otherwise} \end{cases} \tag{5.7}$$

де: $c(A \to \alpha)$ – це кількість разів, коли правило $A \to \alpha$ застосовувалось для побудови дерева розпізнавання навчального набору, $c(A)$ – загальна кількість використаних правил виводу, які мають лівий оператор A.

Параметр ε призначений для врахування відсутніх подій (граматичних конструкцій), які представлені у граматиці, але відсутні у виразах тренувального набору. У разі якщо тренувальний набір недостатньо великий, можуть використовуватися інші методи згладжування замість ε. Лаплас (*Add-One*) згладжування [148] – один із найбільш простих і поширених методів.

5.2. Мовна модель

Модель граматики визначає правила побудови МВ та їх ймовірності з підвиразів. Але це призводить до того, що ігнорується інформація про символи, які включені в ці підвирази. Для прикладу, у побудові МВ, враховуючи тільки ймовірності правил виводу, наступні вирази будуть мати однакові ймовірності: a^{+b} та $a^{\pm b}$; $\sum_{n+m}$ та $\sum_{\eta+\mu}$ тощо.

У цій роботі для врахування символів пропонується застосувати біграми як одну з найуспішніших мовних моделей [35]. Часто біграми застосовують у завданнях розпізнавання 1D-послідовностей, таких як розпізнавання мови. Основними перевагами такої моделі є її обчислювальна складність $O(1)$ та невеликі вимоги

до обсягу пам'яті. Для того, щоб врахувати 2D-структуру МВ, в цій роботі розраховуються два показники. Перший показник $p_l(B|Ar)$ відповідає за ймовірність символу B після символу A для заданого просторового відношення r та визначається такою формулою:

$$p_l(B|Ar) = \frac{C(A, B|r)}{C(B|r)} \tag{5.8}$$

де: $C(A, B|r)$ – кількість прикладів, де символи A та B послідовно зустрічаються у відношенні r; $C(B|r)$ – кількість прикладів, де символ B зустрічається у відношенні r.

Другий показник $p_l(r|AB)$ визначає ймовірність просторового відношення r між символами A та B:

$$p_l(r|AB) = \frac{C(A, B|r)}{C(AB)} \tag{5.9}$$

де: $C(AB)$ – це кількість разів, де послідовність AB зустрічалась в усіх просторових відношеннях.

5.3. Загальний опис алгоритму розбору

Для побудови дерева розбору МВ використовується розширення КЯК алгоритму для 2D-СКВГ граматик. Цей алгоритм дає можливість обчислити найбільш ймовірне дерево синтаксичного аналізу відповідно до заданої моделі граматики. Вхідними даними алгоритму є результат сегментації та класифікації символів C, який містить інформацію про початкову та кінцеву точки у вхідній послідовності, варіанти класифікації символів та їх ймовірності.

КЯК алгоритм синтаксичного аналізу є методом динамічного програмування, який складається з двох етапів. Перший етап створює для всіх можливих варіантів сегментації та класифікації гіпотези термінальних символів та обчислює їх ймовірності. На другому етапі будуються поєднання різних гіпотез, обчислюється їх ймовірність, і таким чином будується структура МВ. Ймовірність гіпотези позначається заданою формулою [123]:

$$p(l, S, A) = \hat{p}(A \overset{*}{\Longrightarrow} S); \quad l = |S| \tag{5.10}$$

де: $p(l, S, A)$ позначає ймовірності нетермінального символу A, що виводиться з набору символів S довжиною l.

Позначимо вхідні дані X як список з N елементів:

$$X = [\ (B, O_{start}, O_{end}, Z = [\ (L, V)\ ]\ )\ ] \tag{5.11}$$

де: кожен елемент X_i містить інформацію про обмежувальний прямокутник B, індекси початкової O_{start} та останньої O_{end} точок, а також варіанти класифікації символів Z та їх ймовірності $p(\alpha) = p(Z.L) = Z.V$.

Таким чином, список X визначає допустимі варіанти сегментації символів, а $X_i.Z$ визначає варіанти класифікації для кожного варіанта сегментації. Загальна кількість варіантів класифікації символів M дорівнює:

$$M = \sum_{i=0}^{N} |X_i.Z| \tag{5.12}$$

На першому етапі алгоритму розбору розраховується ймовірність гіпотези для термінальних символів як добуток ймовірностей різних елементів та визначається за формулою:

$$p(1, S_{i,j}, A) = \max_{i,j}(p(A \to \alpha) \cdot p(\alpha)) \tag{5.13}$$

де: A – нетермінальний символ заданої граматики; $S_{i,j}$ – набір символів, який включає символ α та $|S_{i,j}| = 1$; $p(A \to \alpha)$ – ймовірність правила виводу символу A, що обчислена за допомогою формули 5.7; $p(\alpha)$ – оцінка ймовірності символу $X_i.Z_j.L$. Створені гіпотези заносяться на перший рівень в таблицю розбору.

На наступному етапі алгоритм створює для кожного рівня нові гіпотези шляхом поєднання наявних гіпотез з попередніх рівнів. Кількість рівнів у таблиці розбору обмежена кількістю варіантів сегментації N. Ймовірність гіпотези розраховується за формулою:

$$\begin{aligned} p(l, s, A) = \ & p(A \overset{r}{\Rightarrow} BC) \cdot p(r|BC) \cdot p_l(C|Br) \cdot p_l(r|BC) \cdot \\ & p(l_B, s_B, A) \cdot p(l_C, s_C, A) \end{aligned} \tag{5.14}$$

де: s – набір символів, який включає символи s_B та s_C, при цьому $s = s_B \cup s_C$, $s_B \cup s_C = \varnothing$, $l = l_B + l_C$; $p(A \overset{r}{\Rightarrow} BC)$ – ймовірність правила виводу символу A (Формула 5.7); $p(r|BC)$ – ймовірність, що отримана за допомогою класифікатора просторових відношень (Розділ 4); $p_l(C|Br)$ – ймовірність символу C після символу B для просторового відношення типу r, що отримана за допомогою моделі мови (формула 5.8); $p_l(r|BC)$ – ймовірність типу просторового відношення r між символами B та C на основі моделі мови (Формула 5.9); $p(l_B, A, s_B, A)$ та $p(l_C, s_C, A)$ – ймовірності гіпотез із попередніх рівнів.

Псевдокод для розбору МВ представлений в Алгоритмі 5.1. Дерево розбору позначено E, де кожен вузол $E(L, s, A)$ представлений списком гіпотез $H = [H_i]$. Кожна гіпотеза містить інформацію про ймовірність p, набір символів s, правило виводу r та вказівники на ліву $left$ та праву $right$ гіпотези. Найбільш ймовірне дерево розбору $H_{\max}$ може бути знайдено в дереві виводу E серед усіх гіпотез довжиною N, для яких правило виводу A належить множині початкових символів заданої граматики:

$$H_{max} = \arg\max_{h} h.p \mid \begin{array}{l} h \in E(N, \cdot, A) \\ A \in G(S) \end{array} \tag{5.15}$$

5.4. Вагові коефіцієнти моделей

У запропонованій для розрахунку ймовірності гіпотези Формулі 5.14 можна виділити дві частини. Перша частина $p(A \overset{r}{\Rightarrow} BC) \cdot p(r|BC) \cdot p_l(C|Br) \cdot p_l(r|BC)$ відповідає за ймовірність події в момент часу t, а друга частина $p(l_B, s_B, A) \cdot p(l_C, s_C, A)$ – за ймовірність послідовності в момент часу $t-1$. Ймовірність події в момент часу t залежить від декількох моделей: моделі граматики, моделі просторових відношень та мовної моделі. Також в цій формулі кожна модель у розрахунку ймовірностей має однакову вагу. Однак це часто призводить до занадто великого впливу мовної моделі, хоча її основною метою є виправляти неточності, пов'язані з класифікацією символів або розміщенням елементів під час написан-

Алгоритм 5.1: Псевдокод алгоритму КЯК для розбору МВ

Input: A list of $N = |X|$ character segmentation and classification primitives $X = [\ (B, O_{start}, O_{end}, Z = [(L_j, V_j)])\ ]$.
Input: A grammar $G = \{N, T, P, S, R\}$
Output: Parse tree E

```
/* Initialization of parse tree E from terminal symbols */
for i = 1 to N do
    for j = 1 to |X[i].Z| do
        α = X[i].Z[j].L ;                          /* symbol label */
        p_α = X[i].Z[j].V ;   /* classification label probability */
        foreach production A → α ∈ G(T) do
            pr = p_α · p(α | A) ;          /* hypothesis probability */
            if H.p > 0 then
                H = new Hypothesis() ;     /* terminal hypothesis */
                H.p = pr;
                H.s = {i};
                H.r = A → α;
                E[1, H.s, A] = E[1, H.s, A] ∪ H;

/* Parse tree construction */
for L = 2 to N do
    for L_left = 1 to L − 1 do
        L_right = L − L_left;
        foreach H_B ∈ E[L_left, ·, ·] do
            foreach H_C ∈ E[L_right, ·, ·] ∧ H_B.s ∩ H_C.s = ∅ do
                foreach production A ⟹ BC ∈ G(P) do
                    p = p(A ⇒
                      BC) · p(r|BC) · p_l(C|Br) · p_l(r|BC) · H_B.p · H_C.p;
                    if p > 0 then
                        H = new Hypothesis() ; /* binary hypothesis */
                        H.p = p;
                        H.s = H_B.s ∪ H_C.s;
                        H.r = A ⇒ BC;
                        H.left = H_B;
                        H.right = H_C;
                        E[L, H.s, A] = E[L, H.s, A] ∪ H;

return E;
```

ня. Для вирішення проблеми занадто великого впливу мовної моделі та проблеми спустошення (*underflow*) пропонується розрахувати вагові коефіцієнти кожної моделі та оперувати в логарифмічній системі. У цьому випадку формули розрахунку ймовірностей (5.13,5.14) можна записати в такому вигляді:

$$\begin{aligned} p(1, S_{i,j}, A) = & \max_{i,j}(K_{term} \cdot \log(p(A \to \alpha)) + K_{char} \cdot \log(p(\alpha))) \\ p(l, s, A) = & K_{bin} \cdot \log(p(A \overset{r}{\Rightarrow} BC)) + \\ & K_{rel} \cdot \log(p(r|BC)) + \\ & K^B_{lang} \cdot \log(p_l(C|Br)) + \\ & K^r_{lang} \cdot \log(p_l(r|BC)) + \\ & p(l_B, s_B, A) + p(l_C, s_C, A) \end{aligned} \tag{5.16}$$

де: K_* – це вагові коефіцієнти кожної моделі. Опис вагових коефіцієнтів наведено в Таблиці 5.1.

Таблиця 5.1. Опис вагових коефіцієнтів моделей для розрахунку ймовірностей у виконанні алгоритму розбору МВ

Ваговий коефіцієнт	Опис
K_{char}	Вплив моделі сегментації та класифікації символів
K_{rel}	Вплив моделі класифікації просторових відношень
K_{term}	Вплив граматичної моделі для правил виводу для термінальних символів $A \to \alpha$
K_{bin}	Вплив граматичної моделі для бінарних правил виводу $A \overset{r}{\Rightarrow} BC$
K^B_{lang}	Вплив показника $p_l(B\|Ar)$ мовної моделі, що відповідає за ймовірність символу B після символу A для заданого просторового відношення r
K^r_{lang}	Вплив показника $p_l(r\|BC)$ мовної моделі, що відповідає за просторове відношення r між символами B та C.

Визначення оптимальних значень вагових коефіцієнтів моделей здійснюється за допомогою генетичного алгоритму [49, 58]. Незважаючи на критику [152] і деякі недоліки, такі як обчислювальна складність та час виконання, даний алгоритм добре підходить для цієї задачі, оскільки розмірність задачі невелика, алгоритм

стійкий та розпаралелюється. Алгоритм 5.2 містить псевдокод генетичного алгоритму для пошуку вагових коефіцієнтів моделей.

Алгоритм 5.2: Псевдокод генетичного алгоритму для пошуку вагових коефіцієнтів моделей

Input: S - train dataset of PMB.
Output: $K = (K_{char}, K_{rel}, K_{term}, K_{bin}, K^B_{lang}, K^r_{lang}) - weights$
`/* Generation 0 */`
$z = 0$;
P_z = a populcation of n-randomly-generated individuals;
`/* function` $fitness$ `estimates recognition accuracy */`
$F = fitness(i)$ for each $i \in P_z$;
$F_{best} = \max F_i$;
$K = \arg\max_{P^i_z} F_i$;
repeat
  `/* 1. Selection */`
  select $(1 - \chi) \times n$ members and insert into P_{z+1};
  `/* 2. Crossover */`
  select $(\chi) \times n$ members of P_z and pair them up;
  produce offsping and insert the offsping into P_{z+1};
  `/* 3. Mutate */`
  select $\mu \times n$ members of P_{z+1} and perform random mutations;
  `/* 4. Evaluate` P_{z+1} `*/`
  $F = fitness(i)$ for each $i \in P_{z+1}$;
  $F_{max} = \max F_i$;
  if $F_{max} > F_{best}$ **then**
    $F_{best} = \max F_i$;
    $K = \arg\max_{P^i_{z+1}} F_i$;
  `/* 5. Increment */`
  $z = z + 1$;
until $z < Z_{\max}$;
return K;

Необхідно відзначити, що навчання коефіцієнтів має відбуватися кожного разу після підготовки нових моделей (класифікації символів, просторових відношень, граматичної або мовної моделей). Але перенавчання не є обов'язковим, якщо зміни моделей були незначними. Прикладом незначних змін може слугувати дода-

вання кількох правил до граматичної моделі або донавчання (*fine tuning*) моделі символів за допомогою невеликого додаткового набору даних. У цьому разі вплив різних моделей практично не змінюється.

Задача пошуку оптимальних коефіцієнтів вимагає додаткового набору даних. При цьому бажано виключити зразки, які брали участь у навчанні моделей сегментації та класифікації символів, або моделей просторових відношень. Використання одних і тих же прикладів може привести до перенавчання системи в цілому. Застосування для навчання коефіцієнтів окремого набору зразків може виступати як додатковий метод регуляризації та позитивно впливати на стабільність системи.

5.5. Мінімізація обчислювальної складності

Представлений алгоритм розбору МВ на кожному кроці зберігає усі гіпотези $E(l, s, A)$ довжиною l, які складаються з набору символів s та забезпечують вивід A. Якщо зберігати тільки гіпотези з найбільшою ймовірністю (жадібний алгоритм), то часова складність алгоритму буде складати $O(N^4 \cdot |G(P)|)$. Недоліком такого підходу є те, що правильні гіпотези можуть бути відкинуті занадто рано. Представлений алгоритм зберігає усі можливі шляхи побудови дерева МВ, тому його складність дорівнює $O(N! \cdot |G(P)|)$. Для мінімізації часу роботи алгоритму розбору в цій роботі запроваджена комбінація з декількох прийомів.

5.5.1. Променевий пошук для скорочення кількості гіпотез

Для оптимізації алгоритму побудови дерева розбору та уникнення помилок «короткозорості», які властиві для жадібних алгоритмів, в цій роботі буде використовуватися променевий пошук.

Означення 5.8. Променевий пошук – це евристичний алгоритм пошуку, який досліджує граф шляхом розширення найбільш перспективних вузлів. На кожному етапі побудови дерева розбору він генерує всі гіпотези, але зберігає тільки певну кількість (так звана ширина променя) найкращих гіпотез.

Променевий пошук також не гарантує пошуку найкращої гіпотези, проте він зменшує алгоритмічну складність та суттєво знижує вимоги до необхідної кількості пам'яті.

Нехай W – ширина променя. Тоді кількість гіпотез у кожному вузлі дерева розбору не має перевищувати ширину променя $|E(l, s, A)| <= W$. Ширина променя W буде визначатися функцією $W = w(L, N, |E_{L-1}|)$, яка залежить від поточного рівня L, кількості рівнів N та кількості гіпотез на попередньому рівні $|E_{L-1}|$. Ширина променя задається такою формулою:

$$w(L, N, |E_{L-1}|) = \begin{cases} 3 + L, & \text{if } L <= 3 \\ min(W_{max}, max(W_{min}, W_{max} + L - N - \frac{|E_{L-1}|}{K_{dam}}), & \text{otherwise} \end{cases} \quad (5.17)$$

де: K_{dam} – коефіцієнт затухання, W_{min} та W_{max} – мінімальна та максимальна ширина променя.

Така функція забезпечує динамічне регулювання ширини променя та не допускає факторіального зростання часу виконання алгоритму. Більше того, часова складність алгоритму наближається до $O(N^4 \cdot |G(P)|)$, але при цьому зменшена ймовірність того, що алгоритм не знайде найкращого розв'язку. Також такий підхід дає можливість реалізувати інтерфейс, який надає користувачеві зробити вибір із кількох варіантів з максимальною оцінкою. Приклад роботи променевого пошуку зображено на Рисунку 5.1.

5.5.2. Регіони пошуку

Альваро у своїй роботі [123] запропонував обмеження простору пошуку для зменшення складності. Головна ідея обмеження простору пошуку полягає в тому, що не потрібно об'єднувати гіпотезу H_B з усіма гіпотезами $H_C \in E[L_{right}, \cdot, \cdot]$. Для кожної гіпотези H_B і просторового відношення r будуються прямокутні регіони пошуку гіпотез, у яких мають бути розмішені усі елементи гіпотези H_C. Такий метод забезпечує суттєве зменшення складності алгоритму розбору, але

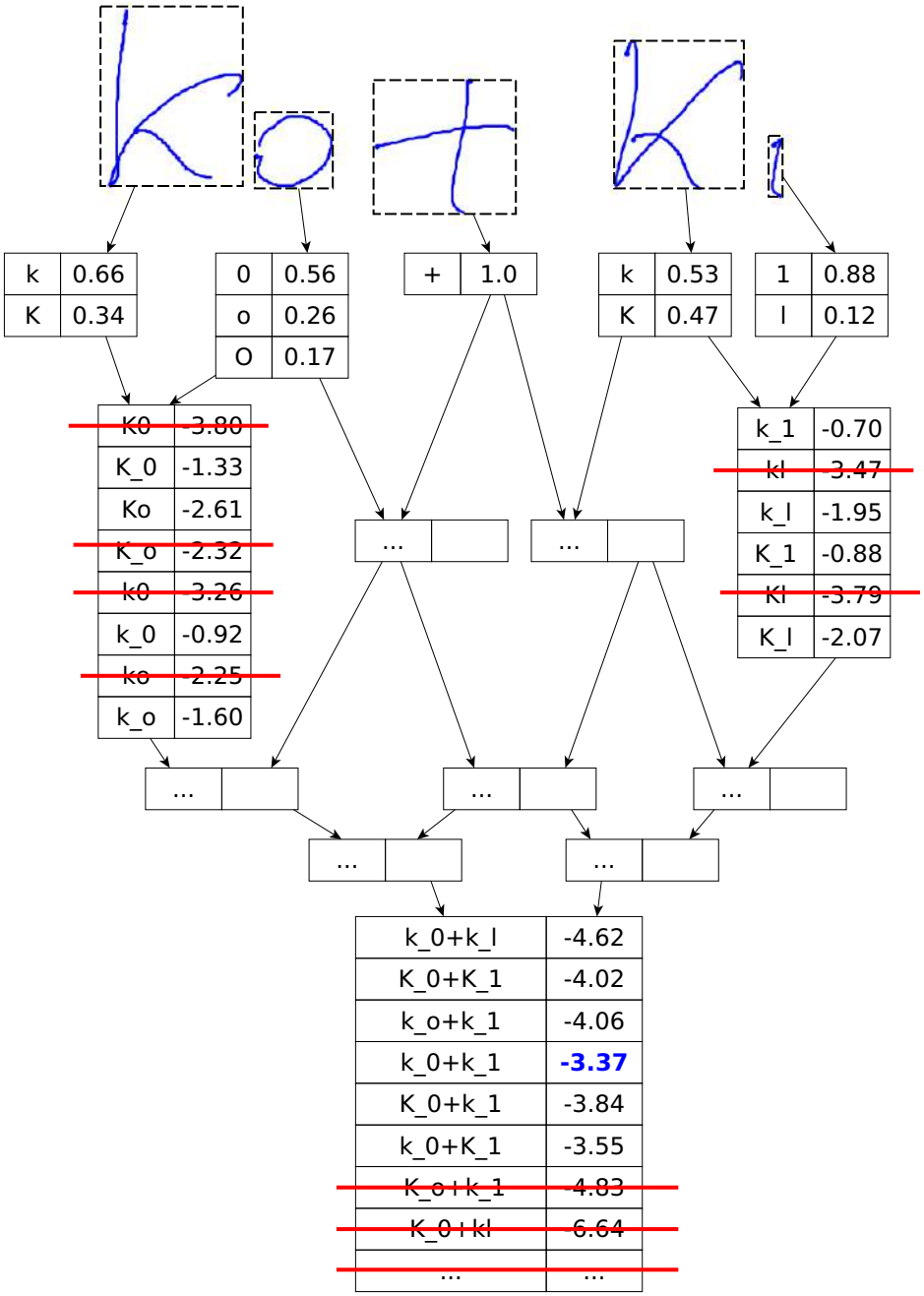


Рис. 5.1. Приклад роботи променевого пошуку у побудові дерева розбору

він має ряд обмежень у розпізнаванні РМВ і більше підходить для друкованих виразів. Одним із недоліків є жорсткі вимоги до елементів з вертикальним розташуванням. Для більш точного пошуку кандидатів для об'єднання пропонується здійснювати за допомогою двох регіонів, де перший (лівий) регіон $\mathcal{R}^l$ встановлює вимоги до лівої границі, а другий (правий) регіон $\mathcal{R}^r$ – до правої границі гіпотези H_C.

Означення 5.9. Область гіпотези $R(H) = (l, r, t, b)$ в 2D-просторі – це мінімальний обмежувальний прямокутник, який містить усі елементи гіпотези H, де (l, t) – ліва верхня, а (r, b) – права нижня координати області.

Рисунок 5.2 демонструє приклади визначення областей пошуку $\mathcal{R}$ для гіпотези H_B залежно від типу просторових відношень. Об'єднання гіпотези H_B з гіпотезою H_C відбувається тільки тоді, коли ліва границя гіпотези H_C належить регіону $\mathcal{R}^l$, а права границя належить регіону $\mathcal{R}^l$

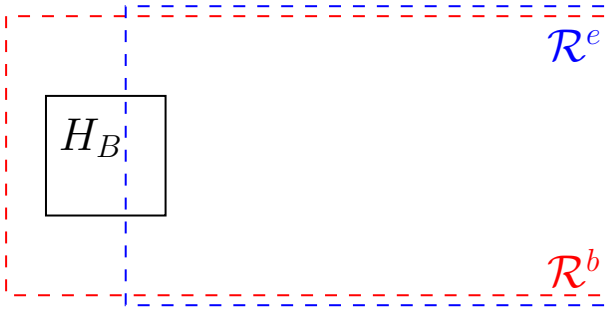
$$R(H_C)(l,t) \in \mathcal{R}^l \vee R(H_C)(l,b) \in \mathcal{R}^l \ \vee \ R(H_C)(r,t) \in \mathcal{R}^r \vee R(H_C)(r,b) \in \mathcal{R}^r \tag{5.18}$$

У разі горизонтальних відношень (*Right*, *Subscript*, *SuperScript*) для побудови регіонів пошуку замість області гіпотези $R(H_B)$ використовується область гіпотези правого символу в рядку $R(rightInRow(H_B))$. Відповідно для перевірки входження в регіони пошуку замість області гіпотези $R(H_C)$ використовується область гіпотези лівого символу в рядку $R(leftInRow(H_C))$. Це пов'язано з тим, що структура лівої або правої частини може включати багаторівневі підвирази. В таких випадках перевірка входження в регіони пошуку по y-координаті неможлива. Тому для горизонтальних відношень застосовуються перевірки тільки для сусідніх символів. Також слід зауважити, що просторові регіони пошуку можуть залежати від оператора або функції, до якої виконується пошук. Область пошуку $\mathcal{R}$ враховує середню ширину R_w та висоту R_h вхідних символів. Для уникнення впливу «великих відхилень», яку можуть спричинити маленькі символи (такі як крапка, кома) та великі оператори (такі як знак дробу та радикалу), такі символи не брали участі у розрахунку середнього розміру символів R_w та R_h.

Як і в запропонованому Альваро варіанті, для ефективного пошуку гіпотез H_C, які потрапляють в область пошуку $\mathcal{R}^l$, після побудови усіх гіпотез довжиною L_A вони сортуються по l координаті області гіпотези. Алгоритмічна складність сортування складає $O(N \log N)$, але це дає можливість виконати бінарний пошук першого кандидата за час $O(\log N)$, а усіх кандидатів – за час $O(N \log N)$. Реалізація

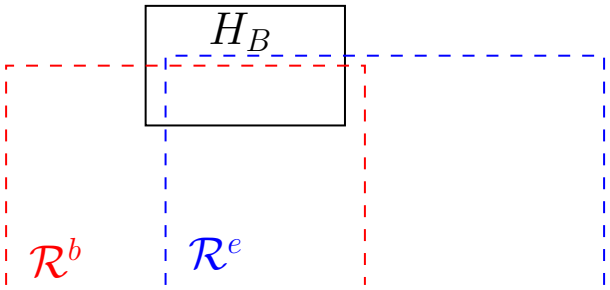
Right, Subscript, SuperScript:

$\mathcal{R}^l.l = r(H_B).l - 0.5 * R_w$
$\mathcal{R}^l.t = r(H_B).t - 2.0 * R_h$
$\mathcal{R}^l.b = r(H_B).b + 2.0 * R_h$
$\mathcal{R}^r.l = (r(H_B).l + r(H_B).r)/2$
$\mathcal{R}^r.t = r(H_B).t - 2.0 * R_h$
$\mathcal{R}^r.b = r(H_B).b + 2.0 * R_h$



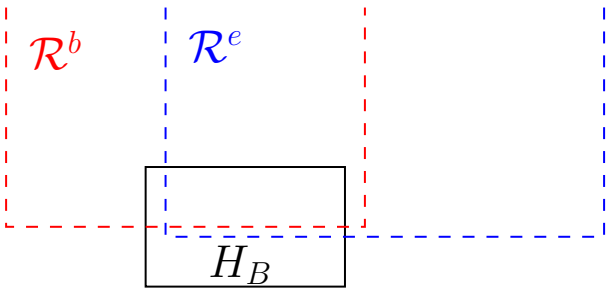
Under:

$\mathcal{R}^l.l = r(H_B).l - 3.0 * R_{avg}.w$
$\mathcal{R}^l.r = r(H_B).r$
$\mathcal{R}^l.t = (r(H_B).t + r(H_B).b)/2$
$\mathcal{R}^r.l = r(H_B).l$
$\mathcal{R}^r.r = r(H_B).r + 8.0 * R_{avg}.w$
$\mathcal{R}^r.t = (r(H_B).t + r(H_B).b)/2$



Over:

$\mathcal{R}^l.l = r(H_B).l - 3.0 * R_{avg}.w$
$\mathcal{R}^l.r = r(H_B).r$
$\mathcal{R}^l.b = (r(H_B).t + r(H_B).b)/2$
$\mathcal{R}^r.l = r(H_B).l$
$\mathcal{R}^r.r = r(H_B).r + 8.0 * R_{avg}.w$
$\mathcal{R}^r.b = (r(H_B).t + r(H_B).b)/2$



Inside:

$\mathcal{R}^l.l = r(H_B).l$
$\mathcal{R}^l.r = r(H_B).r + 1.0 * R_{avg}.w$
$\mathcal{R}^l.t = r(H_B).t$
$\mathcal{R}^l.b = r(H_B).b + 1.0 * R_{avg}.h$
$\mathcal{R}^r.l = r(H_B).l$
$\mathcal{R}^r.r = r(H_B).r + 5.0 * R_{avg}.w$
$\mathcal{R}^r.t = r(H_B).t$
$\mathcal{R}^r.b = r(H_B).b + 3.0 * R_{avg}.h$

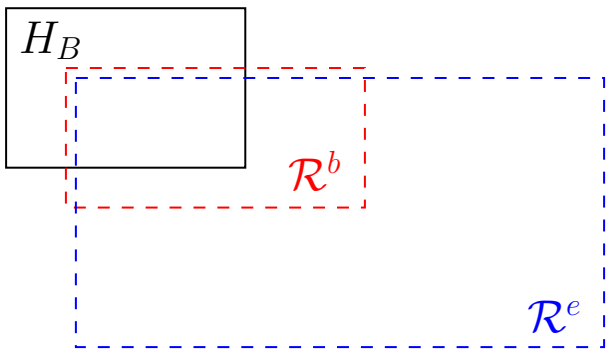
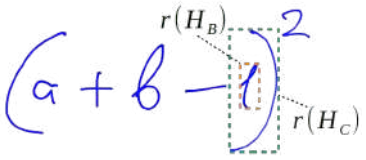


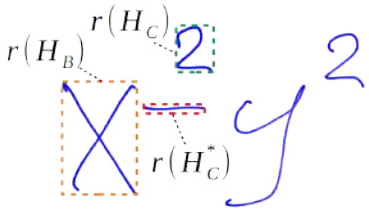
Рис. 5.2. Просторові регіони для пошуку гіпотез відповідно до різних просторових відношень

пошуку може бути заснована на методі дерева регіонів [149, 203], що належить до задач обчислювальної геометрії.

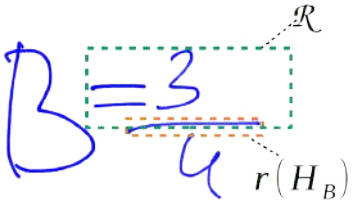
У своїй роботі Альваро враховував тільки першого кандидата з області пошуку. Така реалізація може призводити до відсутності потрібних гіпотез. Це пов'язано з особливістю рукописного вводу, де не завжди дотримуються вимоги до розташування елементів МВ, відстань між елементами залежить від стилю написання та наявності багаторівневих виразів, у яких буває складно чітко визначити правильний рівень розташування елементів (Рисунок 5.3).



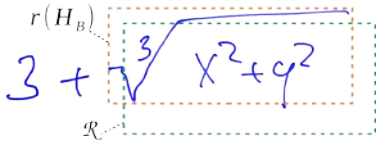
(а) $(a+b-1)^2$. Символ ')' має ліву координату l менш ніж символ '1'



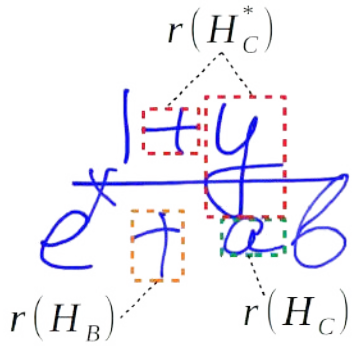
(б) x^2-y^2. Символ '-' має ліву координату l менш ніж символ '2'



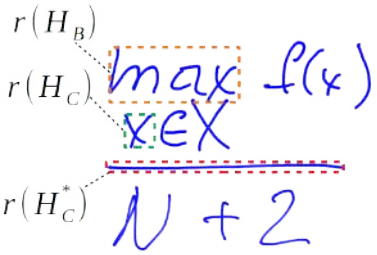
(в) $B=\frac{3}{4}$. Символ '=' міститься в області пошуку для знака дробу і відношення *Over*.



(г) $3+\sqrt[3]{x^2+y^2}$. Ступінь радикалу міститься в області пошуку для відношення *Inside*.



(д) $\frac{1+y}{e^x+ab}$. Символи, які належать чисельнику, знаходяться між сусідніми символами знаменника за координатою $r(H).l$.



(е) $\frac{\max\limits_{x\in X} f(x)}{N+2}$. Символ дробу знаходиться в області пошуку для функції max

Рис. 5.3. Приклади рукописних математичних виразів, де об'єднання тільки з першою гіпотезою за координатою l з області пошуку призводить до побудови неправильного виразу

Тому в цій роботі регіони пошуку значно розширені та допускається об'єднання гіпотези H_B з декількома гіпотезами з області пошуку $\mathcal{R}$. Зрозуміло, що не всі гіпотези з області пошуку $\mathcal{R}$ мають об'єднуватися з гіпотезою H_B. Тому для зменшення кількості гіпотез з області пошуку $\mathcal{R}$ ця область пошуку динамічно зменшується після успішного створення кожної нової гіпотези H_A. При побудо-

ві виразів у горизонтальному напрямку (*Right*, *SubScript* та *SuperScript*) область пошуку $\mathcal{R}^l$ зменшується за такою формулою:

$$\mathcal{R}^l.l = min(R(H_B).r + R_w, R(H_C).l) \tag{5.19}$$

Оскільки математичні вирази переважно мають напрямок зліва направо, то такий механізм є ефективним та дає можливість правильно будувати МВ, у яких не дотримано вимоги щодо розташування елементів (Рисунок 5.3(а), 5.3(б)).

Таким чином, динамічне оновленням області пошуку допомагає уникнути помилок завчасного відкидання правильних гіпотез на етапі побудови МВ. Також це позитивно впливає на складність алгоритму, тому що при горизонтальному напрямку об'єднання гіпотез найчастіше вибирається тільки одна гіпотеза з області пошуку. Для деяких почерків або при недбалому написанні МВ допускається створення декількох гіпотез, де кількість зазвичай не перевищує двох або трьох, але при цьому значно підвищується якість розпізнавання. Але залишається проблема швидкодії алгоритму при вертикальному розміщенні елементів (Рисунок 5.3(г), 5.3(е)) та при наявності радикалів (Рисунок 5.3(д)).

5.5.3. Домінуючі символи

Вперше концепція домінуючих символів була запропонована у роботі [28] для задачі розпізнавання друкованих математичних виразів. У подальшому ідея була застосована для рукописних математичних виразів у роботі [156] для підвищення точності розпізнавання методом побудови мінімального кістякового дерева. Так було запропоновано враховувати домінуючі символи для розрахунку ваг ребер графу.

У цій роботі ідея домінуючих символів адаптована для граматичних методів побудови МВ. Але головною метою застосування домінуючих символів є зменшення складності алгоритму розбору з вертикально розташованими елементами та для елементів із просторовим відношенням *Inside*.

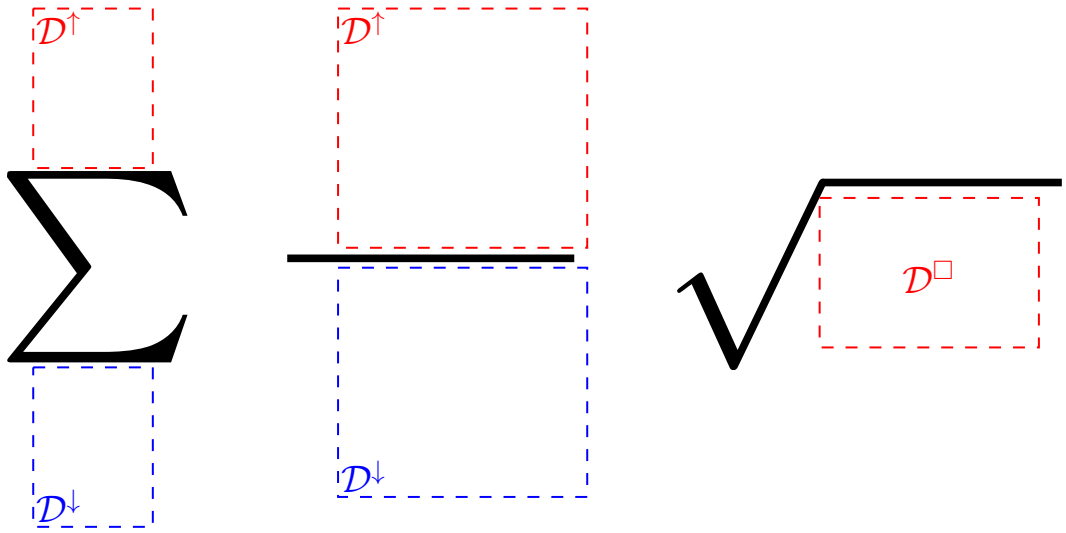
Ми говоримо, що символи домінують над своїми аргументами. Аргументи мають більш низький пріоритет, ніж домінуючий символ.

Означення 5.10. Домінуючий символ – це символ оператора або функції, для яких операнди задаються з використанням вертикальних просторових відношень *Under*, *Over*. Прикладом домінуючих символів є знак дробу, знак радикалу, великі оператори ($\int$, $\prod$, $\sum$), горизонтальні стрілки, деякі функції (arg max, lim).

$$\Gamma = \{\text{frac, radical, large operators, H-arrows, functions with limits}\}$$

Означення 5.11. Символ Ψ домінує над символом ψ, якщо ψ в області домінування символу Ψ.

Означення 5.12. Область домінування $\mathcal{D}^r(H) = (l, r, t, b)$ в 2D-просторі – це прямокутник, який визначає очікуване розташування його операндів. Знак дробу, великі оператори ($\int$, $\prod$, $\sum$) та горизонтальні стрілки мають дві області домінування: зверху $\mathcal{D}^{\uparrow}$ та знизу $\mathcal{D}^{\downarrow}$. Знак радикалу має лише одну область домінування – всередині $\mathcal{D}^{\square}$. Приклади областей домінування зображені на Рисунку 5.4.



(а) Області домінування для $\sum$

(б) Області домінування для знака дробу

(в) Область домінування для радикалу

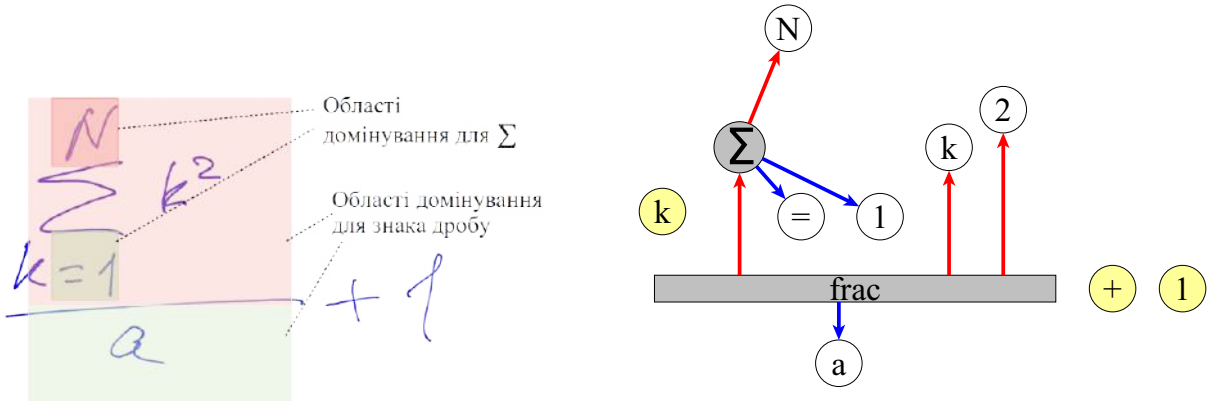
Рис. 5.4. Приклади областей домінування

Як бачимо, області домінування значно менші, ніж області пошуку. Область домінування визначає обов'язковий набір елементів гіпотези H_C в об'єднанні з гіпотезою H_B для побудови гіпотези H_A, використовуючи правило виводу $A \stackrel{r}{\Longrightarrow} BC$. Нехай $\mathcal{S}(H_B, r)$ – функція, що повертає список елементів, які належать обла-

сті домінування $\mathcal{D}(H_B, r)$. Тоді умова побудови гіпотези H_A визначається формулою:

$$\mathcal{S}(H_B, r) \cap H_C.s = \mathcal{S}(H_B, r) \tag{5.20}$$

Очевидно, що одні домінантні символи можуть домінувати над іншими домінантними символами. Це означає, що один символ може потрапити в декілька областей домінування. Як наслідок, це може призвести до неможливості побудови виразу через суперечливі вимоги до належності одного символу. Для уникнення такої ситуації будується дерево домінування, при якому кожен символ може мати тільки один домінуючий символ. Приклад такого дерева зображено на Рисунку 5.5.



(а) Рукописний вираз та області домінування

(б) Дерево домінування

Рис. 5.5. Приклад побудови області домінування та дерева домінування для рукописного виразу $\frac{\sum\limits_{k=1}^{N} k^2}{a} + 1$

Алгоритм 5.3 містить псевдокод побудови дерева домінування. Обробка здійснюється в порядку зменшення довжини символу. Якщо символ ψ потрапляє в область домінування двох або більше символів, то домінантним символом вибирається останній, тобто найменший. Результатом роботи алгоритму є дерево $D = [\ (r, \Psi, \psi)\ ]$, яке представлено у вигляді списку ребер для кожного типу відношень $r \in \{\uparrow, \downarrow, \square\}$. Таким чином, функцію $\mathcal{S}(H_B, r)$ можна представити як множину термінальних символів $H_C.s$, для яких існує відповідне ребро

$(r, H_B.s, H_C.s)$ в дереві домінування D:

$$\mathcal{S}(H_B, r) = \{H_C.s\} \quad | \quad H_C \in E[1, H_C.s, \cdot] \wedge (r, H_B.s, H_C.s) \in D \tag{5.21}$$

Для забезпечення ефективного обчислення умови побудови гіпотез будемо використовувати бітові карти (*bitset*). Так, кожна гіпотеза зберігає в собі бітову карту, яка зберігає набір елементів, що беруть участь у її побудові. А кожна гіпотеза для домінуючого символу зберігає для кожного типу домінуючих відношень r бітову карту символів, які обов'язково мають брати участь в об'єднанні з гіпотезою домінуючого символу і тільки для відповідного відношення. Таблиця 5.2 містить приклад дерева домінування у вигляді бітової карти для Рисунку 5.5.

Таблиця 5.2. Приклад дерева домінування у вигляді бітової карти

	frac	$\sum$	k	$=$	1	N	k	2	a	$+$	1
	0	1	2	3	4	5	6	7	8	9	10
frac $\uparrow$	0	1	0	1	1	1	1	1	0	0	0
frac $\downarrow$	0	0	0	0	0	0	0	0	1	0	0
$\sum \uparrow$	0	0	0	0	0	1	0	0	0	0	0
$\sum \downarrow$	0	0	0	1	1	0	0	0	0	0	0

5.5.4. Регіон охоплення гіпотези

Як уже зазначалося, в РМВ не виконуються вимоги з розташування, та часто області елементів перетинаються. Тому алгоритм розбору використовує динамічну область пошуку з можливістю об'єднання гіпотези H_B з декількома гіпотезами H_C однакової довжини, які містять різні набори символів. Зазвичай кількість «зайвих» гіпотез не перевищує двох або трьох, але такі поєднання можуть призвести до експоненційного росту кількості гіпотез на наступних рівнях алгоритму розбору.

Для уникнення експоненційного накопичення «зайвих» гіпотез пропонується робити перевірку пропущених елементів. Для цього запровадимо поняття «область охоплення гіпотези».

Алгоритм 5.3: Псевдокод алгоритму побудови дерева домінування

```
Algorithm BuildDominance()
    Input: A list of N = |X| character segmentation and classification
           primitives X = [ (B, O_start, O_end, Z = [(L_j, V_j)]) ].
    Output: Dominance tree D = [ (r, Ψ, ψ) ]

    Procedure ProcessAllNodes(nodes, D)
        while nodes ≠ ∅ do
            ProcessNode(nodes[0], nodes, D);
            nodes = nodes \ nodes[0];

    Procedure ProcessNode(Ψ, nodes, D)
        if Ψ.type = 0 then
            return;

        foreach r ∈ R do                                   /* R = {↑, ↓, □} */
            foreach ψ ∈ nodes do
                if ψ.symbol ∈ D^r(Ψ.symbol) then   /* Ψ dominates ψ */
                    D = D ∪ (r, Ψ, ψ) ; /* Extend dominance tree D */
                    Ψ.children^r = Ψ.children^r ∪ ψ;
                    ψ.parent = Ψ;

            /* Build a hierarchy for children */
            ProcessAllNodes(Ψ.children^r, D);
            /* Exclude processed nodes */
            nodes = nodes \ Ψ.children^r;

    /* Initialization of parse tree E from terminal symbols
       */
    for i = 1 to N do
        for j = 1 to |X[i].Z| do
            node = new Node();
            node.type = X[i].Z[j].L ∈ Γ ;          /* Dominant Symbol */
            node.symbol = X[i].Z[j];
            nodes = nodes ∪ node;

    /* Reverse sort by node type and symbol width */
    sort (nodes);

    /* Recursive tree construction */
    ProcessAllNodes(nodes, D) ;

    return D;
```

Означення 5.13. Область охоплення гіпотези $\mathcal{U}(H) = (l, r, t, b)$ – це прямокутник, який включає в себе усі елементи гіпотези, крім крайніх елементів підвиразу в поточній базовій лінії. Якщо крайній правий елемент містить підвираз у вигляді верхнього або нижнього індексу, то ці елементи також не включаються в область покриття гіпотези. Область покриття гіпотези може бути розрахована за формулою:

$$\begin{aligned} \mathcal{U}(H).l &= R(leftInRow(H)).r \\ \mathcal{U}(H).r &= R(rightInRow(H)).l \\ \mathcal{U}(H).t &= R(H).t \\ \mathcal{U}(H).b &= R(H).b \end{aligned} \tag{5.22}$$

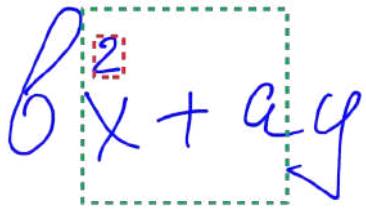
де: $leftInRow(H)$ та $rightInRow(H)$ – функції, що вертають саму ліву та праву термінальні гіпотези в поточній базовій лінії.

Тоді гіпотеза H_A, що будується за правилом виводу $A \overset{r}{\Longrightarrow} BC$, може бути додана до дерева розбору E тільки тоді, коли не існує символу $H_D.s$, який не бере участі в побудові гіпотези $H_D.s \notin H_A.s$ та належить області охоплення гіпотези $R(H_D) \cap \mathcal{U}(H_A) = R(H_D)$. Вимога до побудови гіпотези H_A з урахуванням області її охоплення може бути представлена такою формулою:

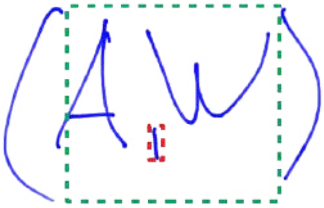
$$\nexists H_D \in E(l, H_D.s, \cdot] \mid H_D.s \notin H_A.s \wedge R(H_D) \cap \mathcal{U}(H_A) = R(H_D) \tag{5.23}$$

Приклади областей покриття гіпотези та пропущених елементів зображено на Рисунку 5.6. Наведені приклади дуже прості, але у випадку, коли вони є частиною більш складного виразу, перевірка області охоплення може значно зменшити час роботи алгоритму.

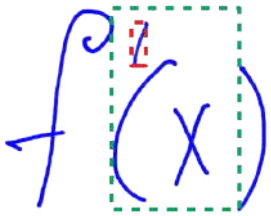
Необхідно відзначити, що перевірка на невраховані символи в області охоплення потрібна тільки у побудові підвиразів в горизонтальному напрямку. Така перевірка є зайвою в побудові виразів у вертикальному напрямку, оскільки для них діють перевірки області домінування. Можна провести аналогію з областю домінування, яка відповідає за вертикальний напрямок. Головною відмінністю є те, що



(а) bx^2+ay. Проміжна гіпотеза $bx+ay$ є «зайвою», оскільки регіон охоплення не містить символу 2



(б) (A, W). Регіон охоплення гіпотези (AW) не містить символу коми



(в) $f'(x)$. Регіон охоплення гіпотези $f(x)$ не включає символу $'$.

Рис. 5.6. Приклади РМВ та «зайвих» гіпотез

область охоплення будується і перевіряється для кожної гіпотези, а не для окремого символу.

5.5.5. Побудова імен функцій

Часто у сегментації та класифікації символів стандартні назви функцій (такі як ‘sin’, ‘lim’, ‘max’ тощо) визначаються окремим символом. Але у разі використання ДДКЧП з функцією втрати НЧК такий підхід продемонстрував гірші показники якості, ніж під час класифікації послідовності з окремих символів. Це призвело до того, що у побудові виразу необхідно будувати імена функцій з окремих символів. Також це збільшує кількість символів на вході алгоритму розбору, що може призвести до підвищення часу відгуку системи. В цій роботі для зменшення часу пропонується розширити граматику 5.24 та створити нові правила виводу ($A \overset{f}{\Longrightarrow} BC$) для побудови імен функцій, а також будувати імена функцій окремим етапом.

$$G = (N, T, P, S, R, F) \tag{5.24}$$

де: F – це множина нетермінальних символів для імен функцій $F \in T$.

Для побудови імен функцій також застосовується правостороння стратегія виводу.

$$SUFFIX\_IN \overset{f}{\Longrightarrow} LETTER\_I \;\; LETTER\_N$$
$$TRIG\_FUNCTION \overset{f}{\Longrightarrow} LETTER\_S \;\; SUFFIX\_IN$$

У разі вдалої побудови імені функції усі символи, які брали участь у побудові гіпотези, блокуються та не можуть бути далі використані для побудови інших підвиразів. Таким чином зменшується кількість символів N та гіпотез, які беруть участь в обробці основного етапу алгоритму КЯК. Оскільки класифікатори символів та просторових відношень надають декілька варіантів класифікації, виникає ймовірність помилкового створення імені функції. Для зменшення кількості таких помилок запроваджується додаткове обмеження на ймовірність класифікації кожного символу $p(\alpha) > \varepsilon_{character}$ та ймовірність просторових відношень між кожною парою гіпотез $p(r|BC) > \varepsilon_{relation}$.

5.6. Висновки до Розділу 5

У цьому розділі дисертаційної роботи представлений метод побудови математичного виразу, де ядром виступає алгоритм розбору та модель граматики математичних виразів, яка задана у вигляді двовимірної стохастичної контекстно-вільної граматики. Алгоритм розбору базується на алгоритмі Кокке-Янгера-Касамі та адаптований під 2D-мови. Для визначення найбільш ймовірного МВ запропонована функція, яка забезпечує інтеграцію моделі сегментації та класифікації символів, класифікації просторових відношень, мовної та граматичної моделей у єдиному показнику. Найважливішою особливістю граматичних підходів є більш висока якість розпізнавання МВ, проте вони є більш вимогливими до обчислювальних ресурсів. Так, складність жадібного варіанта алгоритму складає $O(N^4 \cdot |G(P)|)$, а якщо розглядати усі можливі шляхи побудови МВ, то алгоритмічна складність буде складати $O(N! \cdot |G(P)|)$.

Тому основний внесок у цьому розділі – це набір підходів, які дають можливість істотно скоротити алгоритмічну складність. Перший підхід полягає у використанні променевого пошуку з динамічним регулюванням ширини променя. Це

дозволяє уникнути факторіального часу виконання і разом з тим уникнути недоліків, які притаманні жадібним алгоритмам. Побудова імен функцій дає можливість зменшити розмірність вхідних даних за рахунок раннього детектування імен функцій.

Наступний підхід заснований на регіоні пошуку, який забезпечує мінімізацію складності за рахунок геометричного пошуку. Метод визначає два регіони, які встановлюють обмеження для лівих і правих координат, а також містить динамічне оновлення регіонів залежно від уже опрацьованих варіантів. Такий підхід є більш гнучким і стійким для розташування елементів, яке властиве рукописному вводу. Побудова дерева домінування допомагає знизити складність алгоритму розбору МВ з вертикально розташованими елементами. Регіони охоплення запобігають створенню непотрібних проміжних вузлів у побудові дерева розбору. Запропоновано три методи оптимізації алгоритму розбору, що засновані на геометричному пошуку.

Оскільки застосування регіонів пошуку дає можливість отримати невелику кількість гіпотез, то застосування всіх зазначених методів дає змогу досягти складності $O(N^{3+\varepsilon} \cdot \log N \cdot |G(P)|)$. Також додатково ефективність кожного з цих методів буде розглянуто в Розділі 7 разом з аналізом впливу на якість розпізнавання.

РОЗДІЛ 6

ПІДГОТОВКА ДАНИХ ДЛЯ ТРЕНУВАННЯ МОДЕЛЕЙ

Сучасні методи розпізнавання, засновані на глибокому навчанні, встановили високі вимоги до розміру навчальних даних. Однак такі дані не завжди є загальнодоступними, часто вони містять незначну кількість прикладів або мають обмеження за кількістю класів. Підготовка наборів даних для тренування та тестування моделей – це дужа дорогий процес, займає багато часу та схильний до наявності помилок під час збору та анотації, особливо для задач розпізнавання рукописного вводу. У багатьох задачах, таких як розпізнавання 2D-мов, анотація наборів даних ускладнюється ще великою кількості класів символів та необхідністю анотації просторових відношень між символами. У цьому розділі пропонується новий підхід до автоматичної анотації РМВ. Запропонований ітеративний підхід забезпечує ієрархічну анотацію з використанням моделі розпізнавання на основі ДДКЧП та невеликого анотованого набору даних як відправної точки і забезпечує збільшення алфавіту, поступово покращуючи точність розпізнавання нових класів символів. Такий підхід не передбачає попередньої перевірки зібраного набору даних і включає три основні етапи: навчальні моделі розпізнавання; автоматична анотація за допомогою алгоритмів розпізнавання та порівняння МВ; автоматична перевірка. Ці етапи повторюються до того часу, поки кількість нових автоматично розпізнаних та анотованих зразків не стане достатньо малою. Зразки, які не пройшли автоматичної перевірки, є підозрілими і потребують ручної перевірки або уточнення, що робиться на останньому етапі. В експерименті були автоматично анотовані понад 85% зразків з точністю анотації на рівні символів понад 99%. Результати експериментів продемонстрували економію близько 90% на ручних операціях.

6.1. Проблеми підготовки даних для навчання і тестування

Підходи до глибокого машинного навчання демонструють вражаючі результати у багатьох областях, і, можливо, це пов'язано з наявністю великих наборів даних для розробки та перевірки моделей. Однак підготовка великих наборів даних включає в себе багато кроків і є трудомісткою, дорогою і часто передбачає велику кількість ручної роботи щодо анотації даних. Цей вид діяльності є нудним та монотонним [131] і, отже, становить одну з основних причин помилок під час підготовки наборів даних [51]. У багатьох задачах анотація є складною та може супроводжуватися розбіжностями в думках серед експертів [175]. З огляду на складність і недоліки ручного анотування багато дослідників зосередилися на розробці підходів до автоматичного анотування. Таким чином, було представлено багато методів у сфері автоматичної анотації зображень [29, 124], відео [48], мовлення [54], тексту [45, 66] та рукописних документів [88, 202].

Підготовка анотованого набору даних для онлайн розпізнавання МВ вимагає чималих зусиль, оскільки необхідно враховувати такі аспекти, як 2D-структуру МВ, велику різноманітність математичних символів [19], довжину виразу, кількість учасників, що писали зразки, наявне обладнання, різні стилі письма, репрезентативність вибірки та робочу силу для перевірки й анотування. Анотація математичних виразів складається з декількох рівнів: штрихів, символів та виразу. Ієрархічна структура анотації потрібна для навчання та оцінки різних моделей системи розпізнавання. Сегментація та класифікація символів вимагають анотації на рівні штрихів. Для класифікації просторових відношень вимагається анотація на рівні символів. Оцінка системи в цілому використовує анотацію на рівні виразів. Рисунок 6.1 показує ієрархічну структуру анотації РМВ: призначення оригінальних штрихів символам, встановлення просторових зв'язків між символами і визначення загального виразу у форматі LaTeX. Як правило, InkML формат призначений для опису рукописних наборів даних [173].

На відміну від анотації реальних даних, багато дослідників зосереджуються на створенні штучних даних за допомогою різних методів [69, 134]. Ці підходи

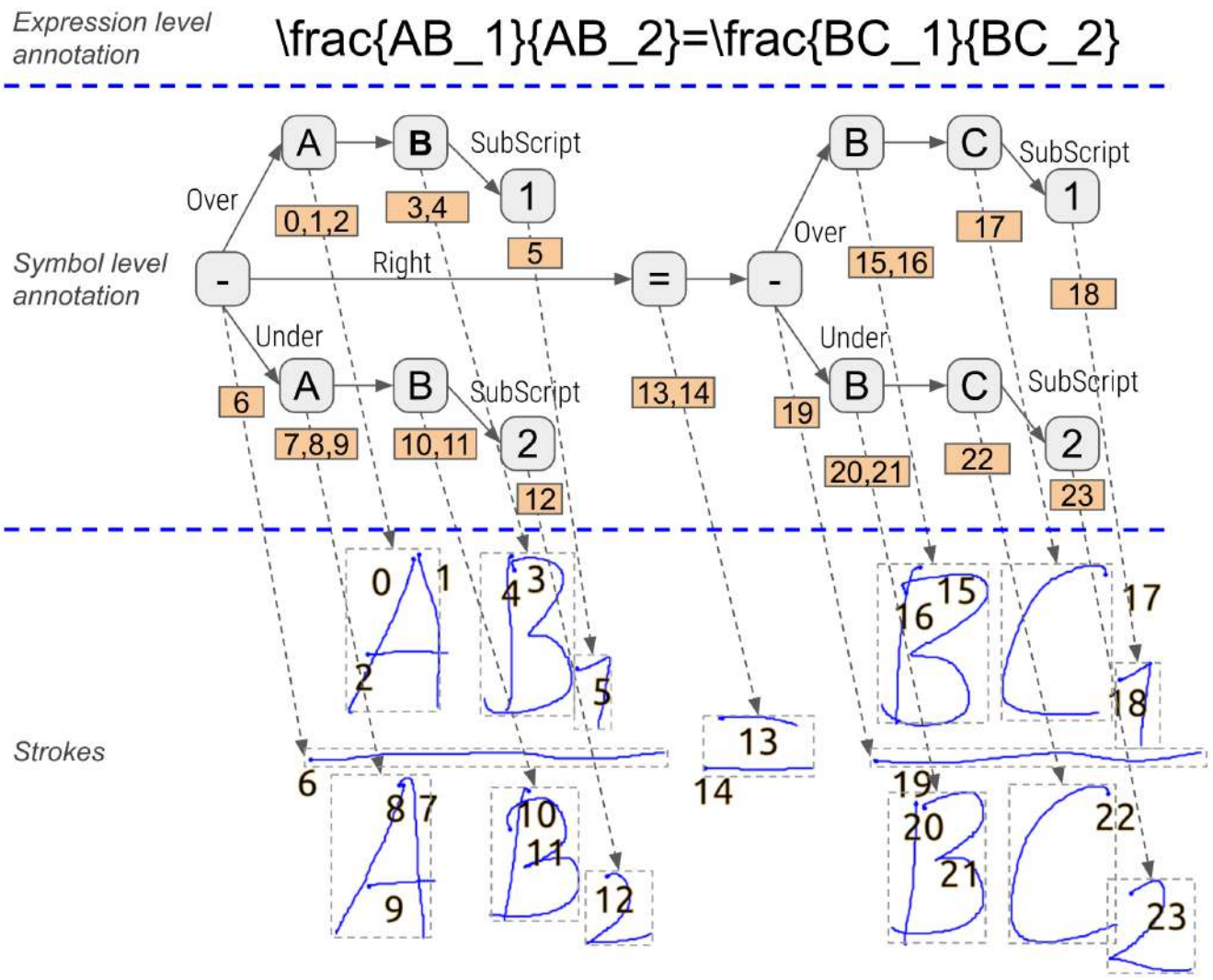


Рис. 6.1. Ієрархічна структура анотації РМВ

можуть сприяти навчанню НМ, але одночасно можуть призвести до несприятливих наслідків у моделях, навчених за допомогою синтетичних даних. У праці [96] автори запропонували кілька стратегій для генерації синтетичних зразків за рахунок невеликого початкового набору даних. Основна ідея полягає у локальних та глобальних спотвореннях. Самі спотворення можуть бути створені за допомогою афінних перетворень та застосування їх до різних елементів МВ, таких як символ, підвираз та цілий вираз. Хоча цей метод показав покращення результатів розпізнавання, кінцевий результат все ще залежить від варіативності стилів рукописного вводу у вихідному наборі даних.

Підхід до пошуку відповідних виразів для створення анотованого набору даних був представлений у роботі [70]. Цей підхід базується на пошуку анотації на

базі наявного корпусу математичних виразів, що містить правильну сегментацію та анотації на рівні символів. Тобто такий підхід може застосовуватися тільки для анотації МВ, для якої існує щонайменше один анотований рукописний приклад. Крім того, цей підхід вимагає попередньо навченої моделі сегментації символів для обробки нещодавно написаних зразків. У праці [109] була запропонована методика анотування МВ, що базується на алгоритмі пошуку відповідності між анотацією на рівні виразу та усіма варіантами сегментації символів та класифікації. Однак цей підхід має високу обчислювальну складність ($O(k^n)$, де k – кількість символів, n – кількість вхідних штрихів) та також вимагає попередньо підготовлених моделей для сегментації та класифікації символів і моделі для класифікації просторових відношень.

Наскрізні рішення перетворюють вхідні дані безпосередньо на математичні позначення, а їх навчальний набір даних вимагає лише анотації на рівні виразів. З іншого боку, для таких рішень потрібні дуже великі навчальні набори даних. Підготовка таких наборів даних здійснюється шляхом генерації синтетичних та доповнених даних на основі наявних анотованих наборів даних. Це, у свою чергу, вимагає детальних анотацій та ретельно перевірених наборів вхідних даних.

6.2. Наявні відкриті набори даних

У 2011 році під час підготовки першого змагання з онлайн розпізнавання PMB (Competition on Recognition of Handwritten Mathematical Expressions або CROHME) Мушаре та співавтори представили новий набір даних, який являв собою об’єднання кількох відкритих наборів даних, таких як MfrDB [154], Mathbrush [90], HAMEX [137], Expressmatch [1] та CIEL [12]. З того часу набори даних, опубліковані як частина CROHME, фактично стали стандартним еталоном для різних досліджень та порівнянь. У межах кожного наступного конкурсу готували новий тестовий набір даних, а навчальний набір розширювали набором тестових зразків з попереднього змагання. Набори даних поширюються у вигляді набору InkML файлів [173], де кожен файл містить таку інформацію: анотація на

рівні виразу у форматі LaTeX та MathML; вхідні штрихи; сегментація та призначені мітки для кожного символу у виразі.

Незважаючи на поступове збільшення, цей набір даних все ще досить малий порівняно з даними для навчання в інших галузях (ImageNet [40] містить понад 14 мільйонів зображень, AudioSet [53] містить понад 2 мільйони звукових кліпів). Особливо це стосується наскрізних РМВ-рішень, які потребують значно більших наборів даних, ніж послідовні та інтегровані рішення. На сьогодні CROHME є основною рушійною силою для розвитку систем розпізнавання МВ як онлайн, так і офлайн. Таблиця 6.1 містить інформацію про розміри CROHME наборів даних.

Таблиця 6.1. CROHME: параметри наборів даних

	2011	2012	2013	2014	2016	2019
Number of symbol classes	56	75	101	101	101	101
Number of isolated symbols in test set	–	–	–	10 061	10 019	15 483
Number of isolated symbols in train set	–	–	–	85 781	85 802	180 440
Number of isolated symbols in validation set	–	–	–	–	10 061	18 435
Number of test MEs	348	488	671	986	1 147	1 199
Number of train MEs	921	1 336	8 836	8 836	8 836	9 993
Number of validation MEs	–	–	–	–	986	986
Number of matrices test MEs	–	–	–	175	250	–
Number of matrices train MEs	–	–	–	362	362	–
Number of matrices validation MEs	–	–	–	–	175	–
Number of writers (test set)	–	–	–	–	50	80
Paper	[119]	[118]	[122]	[120]	[121]	[112]

6.3. Автоматизація процесу анотації

Мета цього етапу – розробити та оцінити підхід до автоматичної або напівавтоматичної анотації РМВ на основі таких основних вимог та припущень:

– наявні анотовані набори даних можуть використовуватися як відправна точка навіть якщо класи не повністю відповідають результуючому набору даних, який потрібно анотувати;
– неможливо спиратися на будь-які попередньо навчені моделі через різний набір класів символів;

– анотацію на рівні виразів можна враховувати для перевірки відповідності структури виразу;
– сумнівні зразки мають бути ідентифіковані для подальшої перевірки вручну;
– має бути можливість розширення списку підтримуваних класів символів порівняно із вхідним набором даних;
– краще не створювати анотацію взагалі, ніж створювати неправильну.

Один із можливих підходів до автоматичної анотації базується на використанні наявної системи розпізнавання. Такий підхід може значно скоротити ручну роботу, пов'язану з підготовкою набору даних. Однак цей підхід накладає обмеження, пов'язані з можливостями цієї системи розпізнавання [109]. Прикладами таких обмежень є розмір алфавіту та граматичні правила. Система розпізнавання також має надавати детальну інформацію про багаторівневу ієрархічну анотацію, включаючи символи та штрихи. Більшість наскрізних рішень не містять усієї необхідної інформації для таких анотацій. Ці обмеження значно звужують перелік прийнятних систем розпізнавання. Після того, як система надасть всю необхідну інформацію для анотації, все ще буде потрібна людина для ручної перевірки створених анотацій.

У цій роботі пропонується підхід до автоматизації анотації за рахунок ітераційного навчання кількох підсистем розпізнавання. На кожному етапі розмір навчального набору збільшується за допомогою зразків, які були анотовані на попередньому етапі. Цей підхід показаний на Рисунку 6.2 і включає такі основні кроки: 1) підготовка початкового анотованого навчального набору; 2) збір нових зразків для анотування; 3) ітераційне навчання моделей та тренувального набору за допомогою автоматичної анотації та перевірки; 4) ручна перевірка нерозмічених або підозрілих зразків. На рисунку суцільні лінії показують переходи між етапами, пунктирні лінії – потоки даних, що пов'язані з навчальним набором даних та зразками для анотації (помаранчеві блоки). По-перше, наявний анотований набір даних включається до тренувального набору для першої ітерації навчання. На другому етапі проводиться збір нових зразків, які поділяються на дві частини:

односимвольні та багатосимвольні. Анотація для односимвольних зразків генерується автоматично, та потім усі односимвольні зразки додаються до навчального набору. Багатосимвольні вирази – це саме той набір, який нам потрібно анотувати. На наступному кроці моделі розпізнавання навчаються на початковому наборі даних, який включає наявний анотований набір та односимвольні вирази. Потім алгоритми розпізнавання виразів та пошуку відповідності здійснюють анотування зібраних зразків. Приклади, які були правильно розпізнані та пройшли автоматичну перевірку, додаються до навчального набору даних. Етапи навчання моделі та автоматичні анотації повторюються, поки збільшення розміру навчального набору даних не стане досить малим ($|AS| < \varepsilon$). Зразки, які не пройшли автоматичну анотацію та перевірку, піддаються верифікації вручну, яка проводиться на останньому етапі.

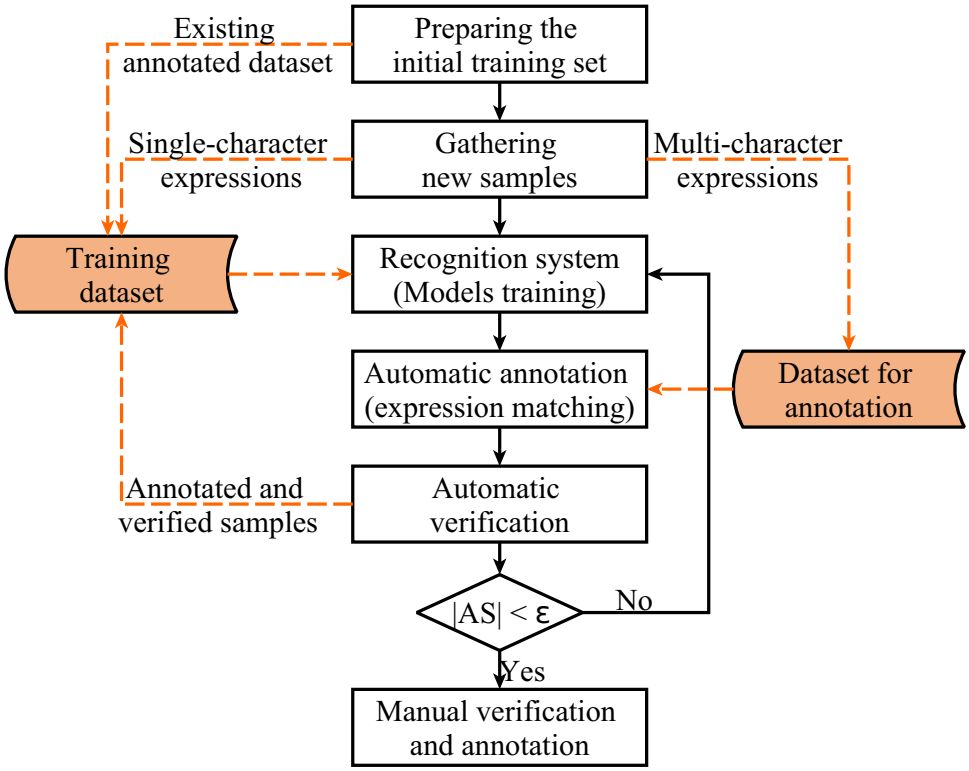


Рис. 6.2. Запропонований підхід до ітераційної анотації

6.3.1. Підготовка початкового навчального набору

Початковий набір даних потрібен для навчання моделі розпізнавання на першій ітерації. Немає особливих вимог до кількості зразків або розміру алфавіту

початкового набору даних. Таким чином, його можна підготувати шляхом ручної анотації частини наявних зразків. Або, наприклад, загальнодоступні відкриті набори даних, такі як CROHME, можна використовувати як початковий набір. Але треба визначити класи символів, які відсутні у початковому наборі порівняно з набором даних, який треба анотувати. Також відсутність зразків конкретних класів символів можна компенсувати за допомогою інших джерел або загальнодоступних наборів даних для розпізнавання рукописного тексту. Забігаючи наперед, зазначимо, що початковий набір даних можна повністю виключити на останніх ітераціях або під час навчання остаточної моделі.

6.3.2. Збір нових зразків для анотування

Учасників попросили переписати запропоновані вирази на мобільних телефонах та планшетах за допомогою стилуса. Самі вирази були зібрані з корпусу математичних виразів Вікіпедії. Запропоновані математичні вирази були переважно засновані на шкільній програмі, містили лише символи алфавіту із 176 символів та були обмежені за довжиною відповідно до розміру екрана. Середня довжина виразу складається приблизно з 12 символів. Щоб збільшити кількість виразів, деякі символи були замінені відповідними символами або послідовністю символів з підтримуваного алфавіту. Наприклад, символ подвійного інтегралу $\iint$ був замінений на два послідовних символи $\int \int$.

Набори виразів були обрані таким чином, щоб усі учасники писали однакову кількість математичних виразів, що містять приблизно однакову кількість символів. Такий процес забезпечує автоматичне отримання анотації на рівні виразів. На додаток до регулярних виразів учасники мали написати односимвольні зразки символів, яких немає у початковому наборі. Односимвольні вирази були додані до початкового набору, щоб компенсувати відсутність класів символів. Для зручності введено такі поняття: базові символи – це символи, які представлені в початковому наборі з багатосимвольних виразів; додаткові символи відсутні в початковому наборі і були додані лише в односимвольних виразах. На цьому ета-

пі немає необхідності перевіряти нові зразки. Тобто вони приймаються такими, якими вони є.

6.3.3. Ітеративна анотація

На цьому етапі ми навчаємо моделі сегментації та класифікації символів та модель просторових відношень на поточному тренувальному наборі даних. Усі зразки, які ще залишилися не анотовані, пропускаються через систему розпізнавання з новими моделями. Алгоритм порівняння виразів (*expression matching*) визначає зразки, які можна автоматично анотувати. Порівняння виразів називають відображення послідовності рукописних вхідних штрихів та символів МВ. Оскільки ми хочемо підготувати набір даних, які не містять неправильних анотацій та потрібні для навчання всіх моделей, анотуємо лише ті приклади, які були розпізнані правильно не тільки на рівні символів, а й на рівні структури. Зразки, які були автоматично анотовані та пройшли автоматичну перевірку, додаються до навчального набору. Ця операція повторюється, поки кількість знову розпізнаних зразків не стане досить малою.

Порівняння виразів. У таких математичних позначеннях, як LaTeX та MathML, один і той самий вираз можна представити кількома способами. МВ представлений за допомогою орієнтованого дерева, де кожен вузол має тип, який визначає відношення, а кінцеві вершини (листя) відповідають символам у виразі. У розпізнаному виразі листя дерева містить інформацію про вхідні штрихи, що відповідають розпізнаному символу. Порядок елементів у цьому дереві чітко визначений. Наприклад, вираз дробу завжди представлений трьома елементами в такому порядку: знак дробу, чисельник, знаменник. Також дерево може містити порожні вузли, які відповідають відсутнім елементам. На Рисунку 6.3 показаний приклад дерева виразу, де для операції суми не задана верхня межа.

Алгоритм порівняння виразів базується на порівнянні дерев. Ця структура є більш точною і представляє відношення між підвиразами, а не між символами. Однак це може створити певні труднощі у розпізнаванні РМВ. Для прикладу, практично неможливо відрізнити такі вирази: x′^2 (x'^2), {x′}^2 (x'^2), x\prime^2 ($x\prime^2$).

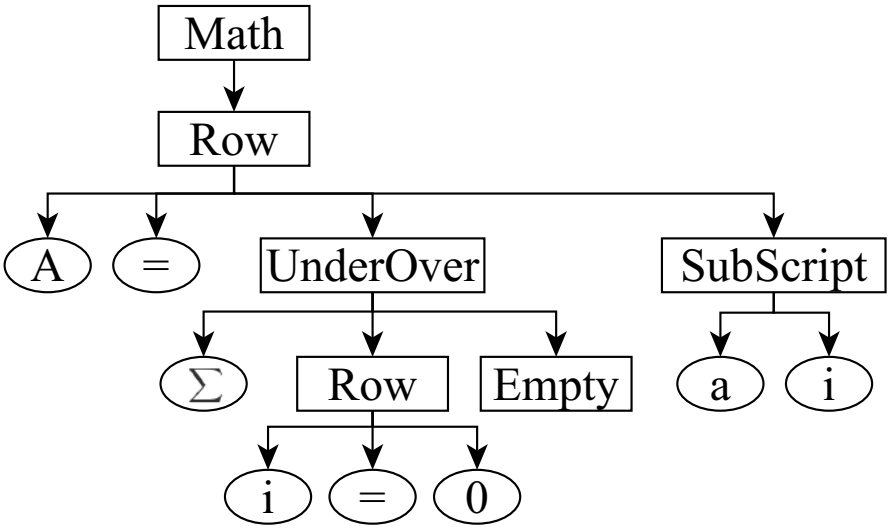


Рис. 6.3. Представлення виразу $A = \sum_{i=0} a_i$ у вигляді орієнтованого дерева

Ці вирази мають різну структуру, але можуть мати однакову семантику. Щоб уникнути помилок під час перевірки структури виразів, усі МВ нормалізуються за такими правилами:

- вузол ‘Row’ має містити більше одного елемента. Якщо вузол ‘Row’ не містить елементів, його можна видалити. Якщо він містить лише один елемент, його можна замінити цим елементом. Таким чином, вираз 2^{^2} (2^{2}) може бути спрощено до вигляду 2^2 (2^2);
- елементи, що відповідають за відстань між символами, не допускаються та можуть бути видалені. Приклад: вираз A_{\;b}($A_{\ b}$) буде трансформований в A_b (A_b);
- відношення ‘SubScript’ і ‘SuperScript’ можна застосувати лише до символу, а не до підвиразу. Якщо ці відношення застосовуються до підвиразу, підвираз із ‘SubScript’ або ‘SuperScript’ приєднується до відповідного символу з основного підвиразу. Для прикладу, вираз {AB_{i+1}}^2 (${AB_{i+1}}^2$) буде трансформовано в AB_{i+1}^2 (AB^2_{i+1});
- елемент знака штриха (prime) має бути у відношенні ‘SuperScript’ ($f\prime \rightarrow f'$).

Однією з основних проблем розпізнавання РМВ є те, що часто неможливо ідентифікувати символ через відсутність контексту. Прикладом такого виразу мо-

же бути $c_1 + c_2$, де практично неможливо класифікувати символи верхнього та нижнього регістру. Деякі вирази у вхідному корпусі можуть містити неправильні символи, де потрібні символи були замінені на подібні. Часто знак градуса замінюють маленькою літерою ‘o’, наприклад: 90^o (90^o) замість 90^\circ (90°). Щоб прискорити процес автоматичного анотування, в цих випадках під час нормалізації виконується заміна усіх таких символів на один. Повний список правил нормалізації символів для автоматичної анотації наведено у Таблиці 6.2. Але таке припущення справедливе лише для перевірки ідентичності виразів для автоматичної анотації та не може застосовуватися для перевірки точності розпізнавання. Також створені анотації на рівні символів містять мітку, яка відповідає вхідному символу у виразі, а не мітку його близнюка.

Таблиця 6.2. Правила нормалізації символів для автоматичної анотації

Вхідний символ				Вихідний символ
0 (digit)	O (capital)	o (small)	∘ (degree/circ)	0 (digit)
1 (digit)	l (small)	| (vline)		1 (digit)
W (capital)	w (small)	ω (omega)		W (capital)
P (capital)	p (small)	ρ (rho)		P (capital)
n (small)	η (eta)			n (small)
K (capital)	k (small)			K (capital)
C (capital)	c (small)	$\subset$ (subset)		C (capital)
X (capital)	x (small)	χ (chi)	$\times$ (times)	X (capital)
S (capital)	s (small)			S (capital)
V (capital)	v (small)	ν (nu)		V (capital)
U (capital)	u (small)	$\cup$ (cup)		U (capital)
M (capital)	m (small)	μ (mu)		M (capital)
B (capital)	β (beta)			B (capital)
Y (capital)	y (small)	γ (gamma)		Y (capital)
. (dot)	, (comma)			. (dot)
: (colon)	; (semicolon)			: (colon)

Якщо система генерує кілька варіантів розпізнавання виразів, то варіанти з меншою оцінкою також можуть брати участь в автоматичній анотації. Отже, для анотації ми перевіряли 5 найкращих варіантів розпізнавання, для кожного з варі-

анта розпізнавання було побудоване дерево, проведена його нормалізація та проведене порівняння з очікуваним деревом.

Автоматична перевірка зразків базується на припущенні, що під час анотування кожен штрих у зразку має належати одному і тільки одному символу. Тобто, якщо штрих не належить жодному символу або належить кільком різним символам, то такий зразок не включається в навчальний набір для наступної ітерації.

6.3.4. Ручна перевірка

Набір для ручної перевірки складається з двох груп зразків. До першої групи належать зразки, які не були автоматично оброблені на попередньому етапі. Для таких прикладів анотація створювалася автоматично на основі результату розпізнавання на останній ітерації. Такий підхід не гарантує правильної анотації, але може значно спростити ручну роботу. Існує кілька причин, чому приклад не міг бути розпізнаний жодною з попередніх моделей: 1) зразок містить додаткові символи або штрихи; 2) один або кілька символів відсутні у зразку; 3) зразок містить неправильний символ ($a + b \rightarrow a + d$); 4) вираз був змінений користувачем, зберігаючи семантику, але з використанням різних символів або відношень ($\frac{1}{2} \rightarrow 1/2$); 5) користувачі неправильно розмістили елементи під час написання ($A_2 \rightarrow A^2$); 6) інші випадки, що пов'язані з різними ситуаціями. Таким чином, більшість випадків ідентифікуються через неправильну анотацію на рівні виразів. У таких випадках рецензент має виправити вираз, і цей приклад можна повторно пропустити через алгоритм для автоматичної анотації. Крім того, у більшості випадків достатньо внести кілька змін, щоб отримати правильну анотацію.

Друга група включає нерозпізнані приклади, які не містять помилок. Такі зразки є джерелом важливої інформації про слабкі місця алгоритму розпізнавання. Крім того, під час ітеративного навчання можна зібрати статистику розпізнавання кожного виразу. Зразки, які були правильно розпізнані лише кілька разів, або ті, які часто потрапляють у топ-N, також можуть містити помилки і вимагати перевірки вручну.

Для ручної перевірки й анотування ми розробили додаток, який фокусується на швидкому виправленні наявної анотації, а не на створенні анотації з нуля (Fig. 6.4). Це дає можливість рецензенту бачити невизначені штрихи та швидко вносити зміни до наявних записів анотацій.

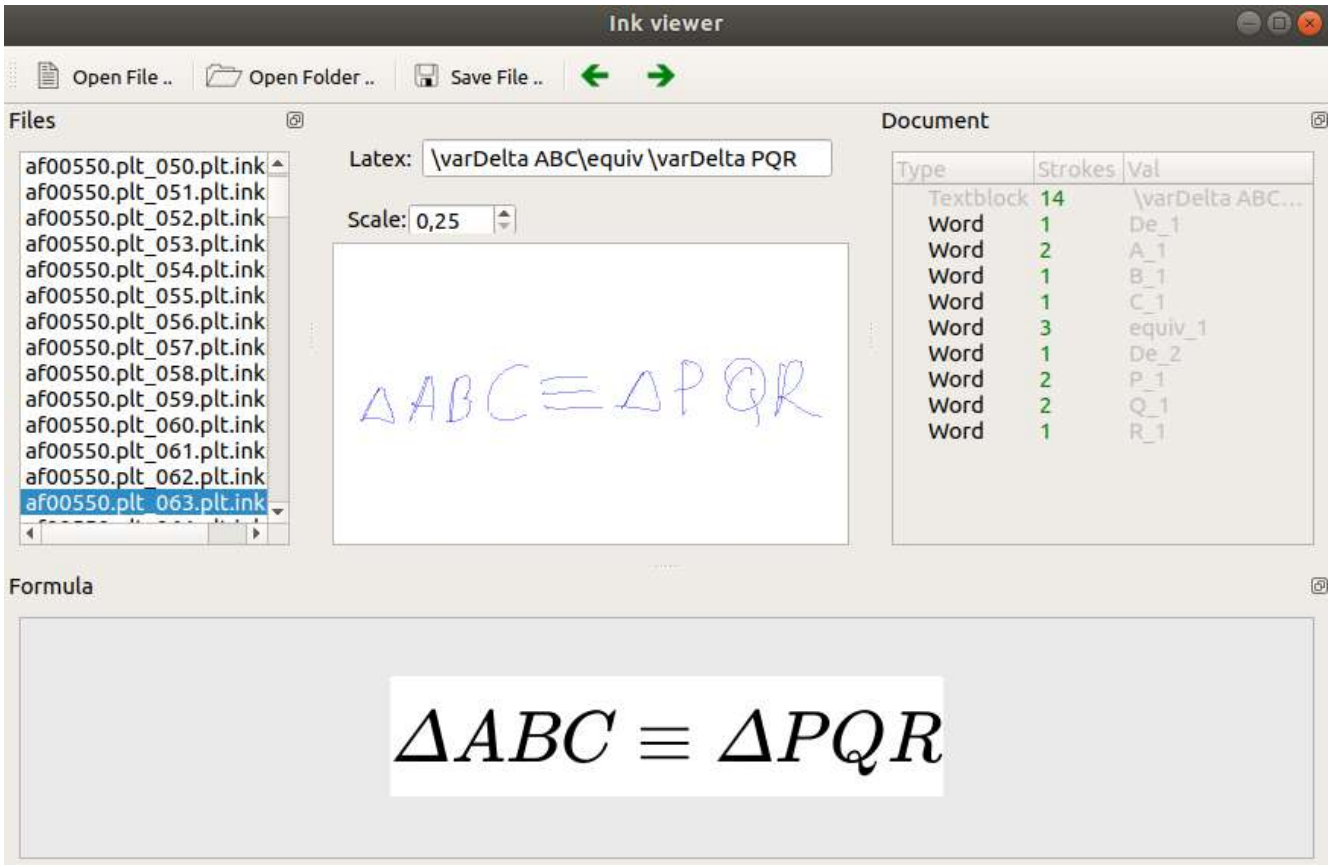


Рис. 6.4. Інтерфейс користувача для ручної перевірки та анотування

6.4. Дослідження ітеративного процесу анотації

Запропонований підхід був перевірений на зібраному неанотованому наборі даних. Цій набір містить 34 014 багатосимвольних виразів (БСВ) та 63 500 односимвольних виразів (ОСВ). Розмір алфавіту цього набору становить 176 символів. Початковий набір даних побудований на основі CROHME [112], щоб продемонструвати здатність запропонованого методу створювати анотації для алфавіту більшого розміру, ніж у початковому наборі даних. Алфавіт набору даних, що потрібно анотувати, містить на 75 класів більше. Для навчання першої моделі ми також додали односимвольні приклади. Цей набір став основою для навчання двох

моделей: сегментації та класифікації символів (модель символів) та моделі класифікації просторових відношень. У Таблиці 6.3 представлена статистика наборів даних.

Таблиця 6.3. Статистика наборів даних для дослідження ітеративного процесу анотації

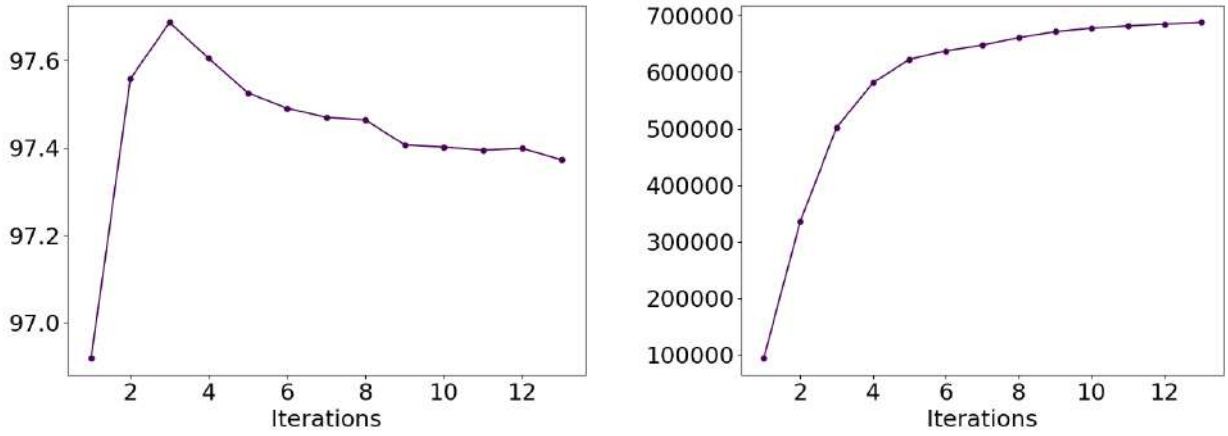
	Початковий НД	Навчальний НД для 0 ітерації	НД для анотації
Розмір алфавіту БСВ	101	101	176
Розмір алфавіту ОСВ	–	75	176
# БСВ	8 508	8 508	34 014
# символів у БСВ	85 584	85 584	431 782
# ОСВ	–	63 500	–
Загальна кількість символів	–	149 084	431 782

6.4.1. Навчання моделі просторових відношень

Як класифікатор просторових відношень використовувався випадковий ліс з однаковими гіперпараметрами для всіх ітерацій. Для першої ітерації було отримано $\approx 95\,150$ зразків просторових відношень з $5\,525$ виразів. Основна мета цього розділу – продемонструвати та перевірити підхід до ітераційної анотації. Тому для автоматичної анотації використовувалась перша відповідна модель і не розглядалися проблеми перенавчання чи недостатньої навченості. Це могло призвести до перенавчання моделі на ранніх ітераціях (Рис. 6.5(а)). Така ситуація істотно не вплинула на швидкість анотації, оскільки перевірялось кілька варіантів розпізнавання виразів. Всього було проведено 13 ітерацій. В результаті 13 ітерацій початковий набір був збільшений до $\approx 687\,700$ зразків з $30\,587$ виразів (Рис. 6.5(б)).

6.4.2. Навчання моделі сегментації та класифікації символів

У результаті 14 ітерацій на етапі автоматичної перевірки було виявлено 382 приклади, які не відповідали вимогам для анотацій. Усі приклади з помилками були виключені з наступної ітераційної підготовки та позначені як кандидати для ручної перевірки. На Рисунку 6.6(а) порівнюються кількості прикладів які були



(а) Точність класифікації просторових відношень

(б) Кількість зразків просторових відношень у навчальному наборі даних

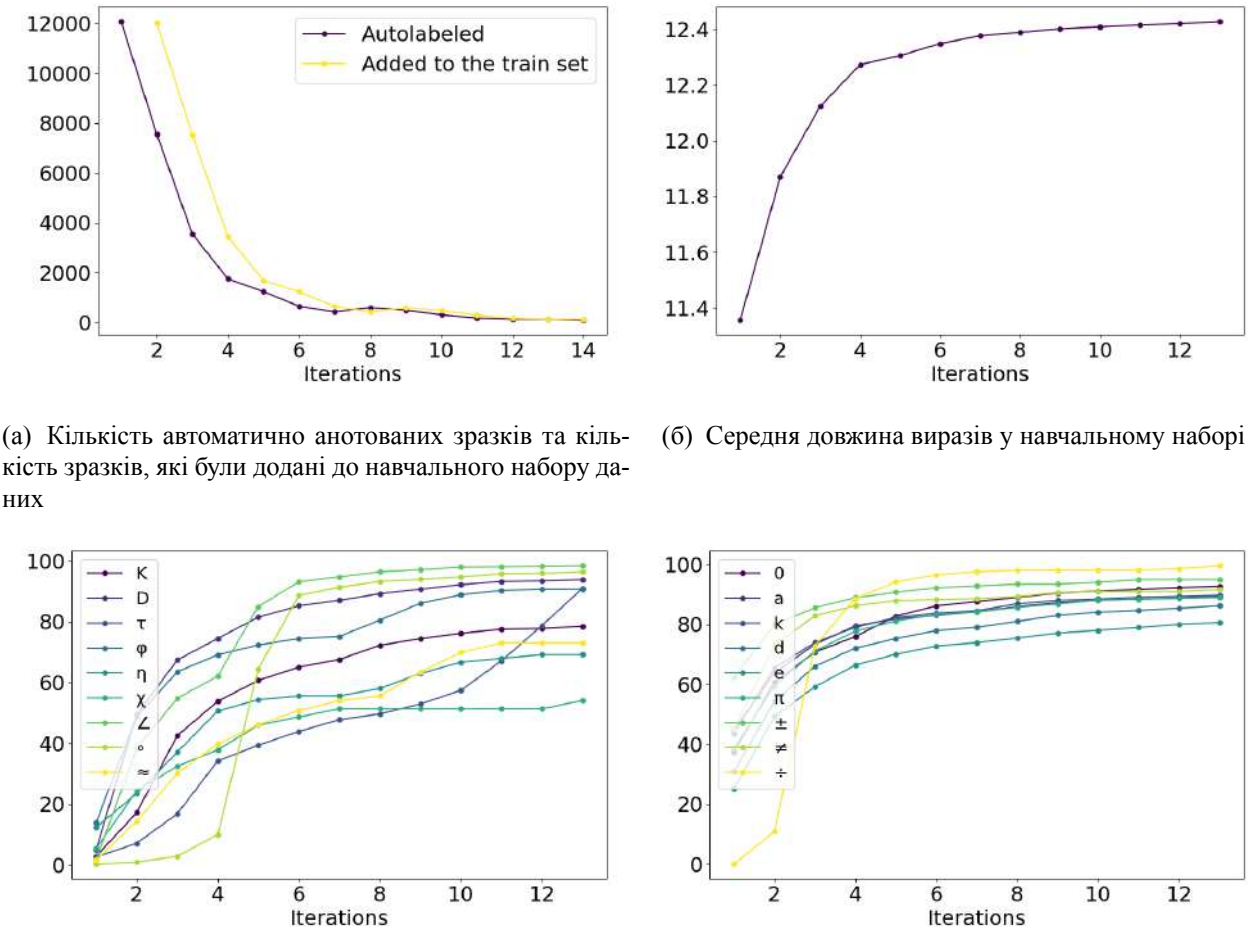
Рис. 6.5. Параметри моделі просторових відношень та розмір набору даних залежно від ітерації анотації

автоматично анотовані, та прикладів, які були добавлені до навчального набору даних.

Найскладніша частина підходу пов'язана з розпізнаванням виразів із символами, які не представлені в початковому багатосимвольному наборі даних. Кількість односимвольних прикладів з такими символами на першій ітерації становила в середньому близько 850. Однак точність розпізнавання таких символів поступово покращується шляхом додавання багатосимвольних виразів до навчального набору. На Рисунку 6.6(в) представлена статистика точності розпізнавання певної підмножини таких символів. Точність розпізнавання базових символів також покращується (Рис. 6.6(г)), але в той же час на перших ітераціях вона вже значно вища, ніж у випадку додаткових символів. Також середня довжина автоматично анотованих виразів збільшується зі збільшенням розміру навчального набору (Рис. 6.6(б)). Середня довжина виразів у початковому наборі складала 10.6, а середня довжина виразу для набору даних анотації становила 12.69.

6.4.3. Аналіз прикладів на етапі автоматичної перевірки.

Перед кожною ітерацією навчання ми автоматично перевіряли всі зразки на наявність явних помилок анотацій. Перевірка проводилася за такими правилами: для кожного символу є мітка з непустим набором штрихів; кожен жест від-



(а) Кількість автоматично анотованих зразків та кількість зразків, які були додані до навчального набору даних

(б) Середня довжина виразів у навчальному наборі

(в) Якість розпізнавання додаткових символів у багатосимвольних виразах

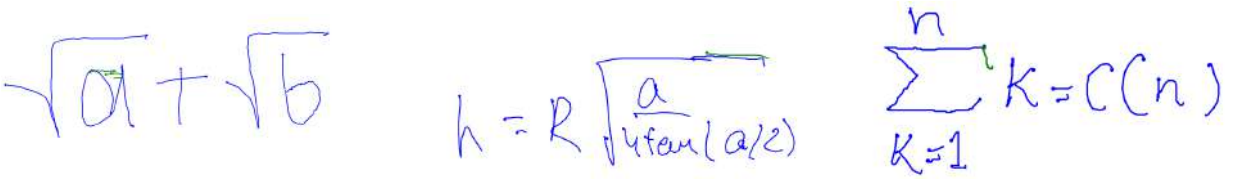
(г) Якість розпізнавання базових символів у багатосимвольних виразах

Рис. 6.6. Параметри моделі сегментації та класифікації символів та результати анотації залежно від ітерації анотації

носиться тільки до одного символу; немає додаткових міток. Ці правила допомагають ідентифікувати найпоширеніші помилки як у ручній, так і в автоматичній анотації. У цьому експерименті усі помилки автоматичної анотації були пов’язані з відсутністю маркування штрихів. Були класифіковані такі причини неправильної анотації: шумні штрихи (Рис. 6.7(а)), штрихи повторювання траєкторії (Рис. 6.7(б)) та відкладені штрихи (Рис. 6.7(в)).

6.4.4. Аналіз прикладів, що були анотовані

За останні ітерації кількість нових зразків становила лише близько 130. На цьому етапі було вирішено, що немає сенсу виконувати наступну ітерацію. Після 14 ітерацій 29 201 вираз було автоматично анотовано. 4 813 (14.15%) зразків зали-



(а) Шумні штрихи (б) Штрихи повторювання траєкторії (в) Відкладені штрихи

Рис. 6.7. Приклади розпізнаних зразків, які не були додані до навчального набору (зелені штрихи вказують на причину неправильної анотації)

шилися без анотації. Зразки без анотацій та зразки, які не пройшли автоматичної перевірки (5 195 виразів), були переглянуті та анотовані вручну. В результаті аналізу зразків, які не було анотовано, визначено три групи. Перша група включала зразки, які вимагали деяких змін на рівні анотації виразу або штрихів, наприклад, видалення шумових штрихів або додаткових символів. Ця група включала 50.76% зразків. Другу групу складали правильні зразки – рукописні дані відповідали анотації на рівні виразу. Однією з найпоширеніших проблем, що запобігали автоматичній анотації зразків у цій групі, була наявність відкладених штрихів. Такі зразки неправильно оброблялися системою розпізнавання. Інша причина пов'язана з особливостями почерку, наприклад, з незвичайним написанням окремих символів. Після ручної анотації та додавання до навчального набору розпізнавання такого стилю почерку покращується. Оскільки автоматично не створювалася анотація до зразків, які містять принаймні одну помилку, вирази з великою кількістю символів також потрапили до цієї групи. Тобто на кожній ітерації автоматичної анотації різні помилки не давали можливості правильно побудувати вираз з оцінкою, достатньою для потрапляння до 5 найкращих варіантів. Остання група включала зразки, які потрібно повністю видалити з набору даних. Кількість таких зразків становить близько 10% від кількості неанотованих зразків, або $\approx$ 1.4% від усіх рукописних виразів. У багатьох вилучених зразках бракувало значної частини рукописного виразу або навіть усього виразу. Ми також видалили зразки, у яких занадто активно використовувалося злите написання. Наприклад, в одному прикладі вираз $\cos(x^2)$ був написаний одним штрихом.

6.5. Точність автоматичної анотації

Точність анотації базується на порівнянні результату автоматичної анотації з ручною. Метрики включають точність для трьох рівнів анотації: вираз, символ та штрих. Точність анотації виразів (*Expression Annotation Recall*, EAR) – це відсоток виразів, які правильно анотовані, тобто всі штрихи віднесені до відповідного символу у виразі. Подібним чином точність анотації символів (Character Annotation Recall, CAR) – це відсоток символів з правильно призначеними штрихами. Точність анотації штрихів (*Stroke Annotation Recall*, SAR) визначає відсоток штрихів, які були віднесені до відповідного символу.

$$EAR = \frac{Number\ of\ correctly\ annotated\ expressions}{Total\ number\ of\ expressions} \tag{6.1}$$

$$CAR = \frac{Number\ of\ symbols\ with\ properly\ assigned\ strokes}{Total\ number\ of\ symbols} \tag{6.2}$$

$$SAR = \frac{Number\ of\ correctly\ annotated\ strokes}{Total\ number\ of\ strokes} \tag{6.3}$$

Для обчислення точності анотації було випадково відібрано 552 ($\approx 1.6\%$) зразки та анотовані вручну. Заборонялося змінювати анотації на рівні виразу, щоб уникнути невідповідності виразів під час порівняння. Метрики точності анотації наведені в Таблиці 6.4.

Таблиця 6.4. Метрики точності анотації

	Кількість об’єктів	Істинно позитивних	Точність (%)
Точність анотації виразів (EAR)	552	546	98.91
Точність анотації символів (CAR)	6885	6879	99.91
Точність анотації штрихів (SAR)	9672	9657	99.84

Помилки анотацій було класифіковано на два типи. Помилка першого типу передбачає невідповідність імен із кількома символами, таких як cot, ln, max, sin тощо. Імена функцій можна анотувати, вказуючи один елемент для всіх символів з назви функції або анотувати кожен символ окремо. Для ручної та автоматичної

анотацій вимагалося встановлювати одну мітку для кожної назви функції. Однак у кількох зразках під час ручної анотації було створено анотації окремих символів. Такі помилки можуть бути віднесені до помилок ручної анотації. Другий тип виявлених помилок пов'язаний із розбіжністю у класифікації шумних штрихів. За ручної анотації такі штрихи були віднесені до найближчого символу або взагалі не мали анотації. У результаті автоматичної анотації такі штрихи іноді пов'язували не з найближчим символом, а з іншим сусіднім символом. Такі помилки трактувалися як помилки автоматичної анотації.

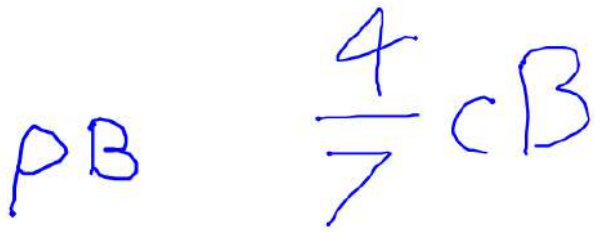
Також був виміряний час, необхідний для ручного анотування одного прикладу, і це приблизно $5 - 6$ хвилин. Крім того, цей час залежить від довжини виразу та навичок анотаторів. Залучення анотаторів із зовнішніх організацій, а також їх навчання потребуватиме набагато більше часу. Також, за нашими спостереженнями, ручна анотація вимагає тривалих перерв на відпочинок. Це можна пояснити монотонністю роботи, яка вимагає постійної концентрації уваги. З такою швидкістю знадобилося б понад 2800 людино-годин (70 людино-тижнів), щоб вручну анотувати зібраний набір ($34\,014$ зразків). Але під час перевірки та виправлення помилок анотації в підозрілому зразку потрібно $1 - 2$ хвилини. Чотири експерти перевірили підозрілі зразки та необроблені зразки менш ніж за тиждень. У запропонованому підході більшість часу витрачено на підготовку моделей для кожної ітерації. Кожна ітерація тривала близько 2 днів, але не вимагала постійного нагляду. Так, на моніторинг процесу та запуск наступної ітерації витрачалося не більше однієї людино-години на добу. Таблиця 6.5 містить порівняльні метрики для ручного та напівавтоматичного анотування.

Цей підхід також може бути скоригований для пришвидшення автоматичної анотації. Найпростіший спосіб зробити це – збільшити кількість варіантів розпізнавання, які враховуються під час перевірки ідентичності виразів. Але це призведе до збільшення кількості зразків з невиявленими помилками. Оскільки ми не проводили первинної перевірки рукописних виразів, однією з основних цілей було виявлення неправильних зразків. Автоматична анотація таких прикладів може призвести до необхідності ручної перевірки всіх зразків.

Таблиця 6.5. Порівняння часу ручного та напівавтоматичного режимів анотацій

	Ручний	Напівавтоматичний
Розмір набору даних	34 014	
# зразків для анотування	34 014	5 195
Час анотування/виправлення 1 зразка	$5-6$ хв.	$1-2$.
Загальний час для анотування	$>2\,800$ год.	180 год.
Кількість ітерацій		14
Тривалість однієї ітерації		46 год.
Контроль однієї ітерації		2 год.

Треба визнати, що не всі спірні вирази були визначені запропонованим способом. Короткі вирази, що складаються з $2-4$ символів, майже завжди мають правильний варіант розпізнавання серед перших N варіантів (Рис. 6.8(а)). Одним із варіантів вирішення цієї проблеми є варіація кількості перевірених кандидатів після розпізнавання залежно від довжини виразу. Це також має зменшити кількість довгих виразів, що не були анотовані. Друга проблема пов'язана з рішенням щодо нормалізації виразу та застосування ознак подібності символів, таких як $Ww\omega \rightarrow w$. У багатьох випадках навіть людині може бути важко правильно ідентифікувати символ. Але ця проблема суттєво зменшується, якщо існує контекст у вигляді розмірів сусідніх символів (Рис. 6.8(б)). Такі приклади також слід ідентифікувати та передавати експертам для остаточного рішення.



(а) ρ_B (б) $\frac{4}{7}CB$

Рис. 6.8. Приклади автоматично анотованих виразів, що містять двозначність та не були ідентифіковані як підозрілі

6.6. Висновки до Розділу 6

У цьому розділі був запропонований метод напівавтоматичної анотації РМВ. На відміну від раніше запропонованих методів, головна перевага цього методу полягає в тому, що він не накладає таких умов, як початкова перевірка зразків або наявність попередньо створених моделей. Основним внеском цієї роботи є методологія підготовки детального анотованого набору даних з розширеним алфавітом порівняно з початковим набором навчальних даних.

Поєднання багатосимвольних та односимвольних виразів дає можливість поступово підвищувати якість розпізнавання нових символів, які не були представлені у початковому наборі. Більше того, цей метод можна використовувати не тільки для створення точної анотації, а й для виявлення тих прикладів, які потребують експертної оцінки. Згідно з дослідженням лише $10-15\%$ зразків потребують втручання експерта. Запропонований підхід також може бути застосований до наборів даних з високим рівнем шуму та для пошуку підозрілих зразків і зразків з помилками. Якість анотацій перевіряли на зразках, що були анотовані вручну. Точність анотації виразів та символів досягла 98.91% та 99.91% відповідно. Понад 85% прикладів РМВ були автоматично анотовані. В результаті обсяг ручної роботи зменшився більш ніж на 90%.

Цей же принцип може бути застосований до підготовки анотованих наборів даних для онлайн/офлайн скетчів, музичних нот, рукописного тексту та багатьох інших завдань, де слід визначати не тільки мітки класів, а й групи символів та відношення між ними. Точність моделей розпізнавання може вплинути на кількість ітерації, але запропонована ітераційна модель напівавтоматичної анотації може спиратися на інші методи та архітектурні рішення систем розпізнавання.

РОЗДІЛ 7

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

У наш час, купуючи мобільний пристрій, такий як смартфон, користувач одержує величезну кількість функцій. При цьому дедалі більше таких функцій реалізовується за допомогою НМ. Кожна з НМ вимагає значного місця в постійній пам'яті. Все це призвело до того, що користувачеві залишається все менше і менше вільного місця під особисті потреби. А збільшення розміру пам'яті веде до істотного подорожчання пристрою. Така експансія нейронних мереж у мобільних пристроях веде до того, що виробники мобільних пристроїв змушені ставити жорсткі вимоги не тільки до швидкості роботи, але і до розміру пам'яті.

Цей розділ містить результати експериментів щодо оцінки роботи системи розпізнавання РМВ включно з аналізом аспектів роботи на мобільних пристроях. Особлива увага приділяється аналізу швидкості роботи системи та впливу запропонованих методів як на якість, так і на швидкість роботи. В Секції 7.1 розглянуто питання, пов'язані з метриками оцінювання. Секція 7.2 містить загальний опис обчислювальних експериментів. Оцінки якості класифікації та сегментації символів наведено у Секції 7.3 з урахуванням запропонованого алгоритму підвищення точності сегментації символів за допомогою додаткової НМ. Секція 7.4 присвячена вивченню ефективності запропонованих методів оптимізації обчислювальної складності граматичних методів структурного аналізу. Результат аналізу швидкодії роботи на мобільних пристроях наведено в Секції 7.5. Наприкінці (Секція 7.6) представлено порівняння з іншими аналогічними системами, спираючись на наукові джерела та результати відкритих змагань.

7.1. Метрики для оцінки систем розпізнавання РМВ

Оцінка рукописних систем розпізнавання МВ – це нетривіальне завдання. Існує ряд проблем, спільних для всіх типів систем розпізнавання, таких як створення або вибір репрезентативних наборів даних та метрик. Розпізнавання МВ має свої труднощі та особливості в процесі оцінки, які пов'язані з 2D структурою МВ, широким спектром класів символів, неоднозначністю математичних позначень та ін. Прикладом неоднозначності в нотації LaTeX є той факт, що один і той самий вираз можна представити різними способами. Так, дріб можна представити за допомогою команд \frac та \over ('a \over b' or '\frac{a}{b}'). Декілька робіт були присвячені поглибленому аналізу проблем оцінки системи розпізнавання МВ [14, 92].

Вибір показників надто залежить від цілей та методу розпізнавання. Зазвичай для оцінки систем розпізнавання РМВ застосовується точність розпізнавання виразів (ТРВ). В англомовній літературі ця метрика називається *Expression rate* (ER). Цей показник визначається як відсоток правильно розпізнаних МВ, тобто які відповідають основній правді з точністю до символів, відношень та структури виразу:

$$ER = \frac{Number\ of\ correctly\ recognized\ expressions}{Total\ number\ of\ expressions} \tag{7.1}$$

Метрика ТРВ призначена для оцінки рішення в цілому, але вона не дає інформації для виявлення слабких місць системи. Іншою інтегрованою метрикою є точність розпізнавання дерева виразу (ТРДВ) або Structure Rate (SR). Він вимірює відсоток виразів, де дерево МВ було розпізнано правильно, ігноруючи мітку символів у вузлах дерева.

$$SR = \frac{Number\ of\ correctly\ recognized\ structures}{Total\ number\ of\ expressions} \tag{7.2}$$

Найпоширеніші показники були запропоновані на етапі панування послідовних рішень і, крім ТРВ та ТРДВ, включають точність сегментації (Character

Segmentation Rate), точність класифікації символів (Character Classification Rate) та точність структурного аналізу [26]. Кожен із згаданих показників безпосередньо пов'язаний з певним кроком розпізнавання. Розвиток методів розпізнавання, складність порівняння, двозначність позначень призвели до створення набору метрик та методів оцінки, які можна класифікувати таким чином: *графові методи* – відповідність між основною правдою та результатом розпізнавання здійснюється за допомогою порівняння графів виразів; *текстові методи* – порівняння здійснюється на основі текстового представлення; *візуальні методи* – оцінка базується на порівнянні двох згенерованих зображень для основної правди та результату розпізнавання. Класифікація та короткий опис показників оцінки представлені в Таблиці 7.1.

Зараз на практиці використовуються тільки графові метрики, які дають можливість визначити помилки на етапі класифікації символів, класифікації відношень або структурні помилки. Далі для можливості порівняння результатів з іншими роботами будемо спиратися на такі метрики:

– точність розпізнавання виразів (ТРВ);

– точність розпізнавання дерева виразу (ТРДВ).

Для більшої зручності та кращого розуміння системи також буде наведено *Top-N* точність. Така точність розраховується шляхом перевірки N варіантів розпізнавання з найбільшою оцінкою. Якщо один із них відповідає основній правді, то метрика враховує зразок як правильний.

7.2. Постановка обчислювальних експериментів

Тренування моделей здійснювалося на закритих наборах даних, які належать компанії «Самсунг». Розмір алфавіту математичних символів становить 176 символів. Перевірки якості розпізнавання здійснювалися на відкритих наборах даних CROHME 2014, 2016, 2019. Розмір алфавіту в відкритих наборах даних становить 101 символ. Для навчання моделі сегментації та класифікації символів використо-

Таблиця 7.1. Класифікація методів оцінки ефективності розпізнавання

Відповідна структура	Примітив	Опис
Графові	Символ	EMERS [141] – це метод оцінки ефективності на основі вартості редагування відстані між двома деревами. Об'єкти – це вершини та ребра дерева. У роботі [3] був запропонований метод порівняння дерев, який використовує додаткове дерево, еквівалентне дереву істинній правді. Основною метою цього методу є врахування особливостей МВ, коли вирази мають різну структуру, але семантично еквівалентні.
	Штрих	Занібі [187] запропонував підхід до оцінки на основі графів, де вершини графу – це вхідні примітиви (штрихи). Такий підхід дає змогу вимірювати усі показники, пов'язані з сегментацією та класифікацією символів і структурним аналізом. Запропоновані метрики представлені як відстань Геммінга. Але у запропонованому методі незначні помилки сегментації або помилки анотації, які не вплинули на кінцевий результат розпізнавання, впливають на інтегровану метрику.
Текстові	Символ	Відстань Левенштейна часто обчислюється для вимірювання точності під час роботи з одновимірними елементами, такими як текст. У праці [89] був представлений спосіб кодування МВ у текст.
Візуальні	Піксель	У роботі [8] був запропонований метод оцінки глобальної помилки на основі порівняння зображень (IMEGE). Цей підхід допомагає виявляти неправильно розпізнані зони, але не дає можливості виявляти вузькі місця під час тестування.

вувалася бібліотека RNNLIB[1]. Інтеграція отриманої моделі НМ здійснювалася за допомогою внутрішньої бібліотеки. Для забезпечення порівняння результатів з іншими рішеннями була підготовлена окрема модель граматики з розміром алфавіту 101 символ. Точність розпізнавання виразу наведена для двох моделей з різним розміром алфавіту.

[1]Alex Graves. RNNLIB is a recurrent neural network library for sequence learning problems. https://sourceforge.net/projects/rnnl/

Вплив окремих елементів на точність або швидкодію системи вивчався методом абляцій. Таке дослідження дає можливість вивчати продуктивність системи шляхом видалення певних компонентів, щоб зрозуміти внесок окремого компонента в загальну систему.

Для обчислення точності та швидкісних характеристик було розроблено консольний додаток, який дає можливість отримати основні метрики. В результаті кожного запуску формується детальний звіт по кожному зразку в тестовому наборі даних, який містить результати розпізнавання та час окремих операцій. На екрані відображається тільки загальна інформація. Приклад результату роботи додатка зображено на Рисунку 7.1. Реалізація додатка з консольним інтерфейсом значно спрощує підтримку декількох операційних систем. Також слід зазначити, що точність розпізнавання не залежить від операційної системи або процесора. Відносний вплив різних елементів системи на швидкість розпізнавання приблизно однаковий на всіх сучасних комп'ютерах, включаючи мобільні платформи. Тому для обчислення усіх метрик та дослідження шляхом абляцій здійснювалося на ноутбуці з процесором *Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz* під операційною системою *Ubuntu 18.04.6 LTS*. Однак абсолютні показники швидкості були отримані на мобільному телефоні *Samsung Note 10* з процесором *Samsung Exynos M3 @ 2.70GHz* на операційній системі *Android 10*.

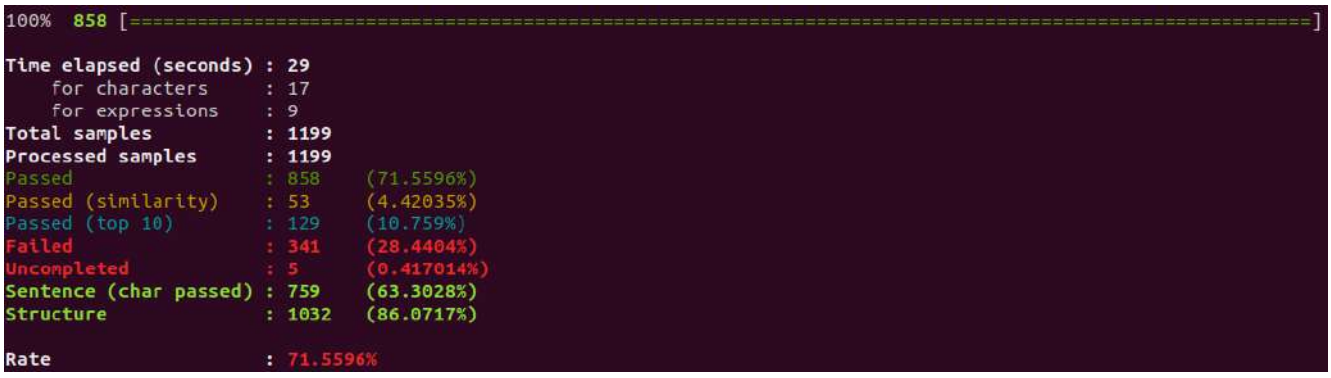


Рис. 7.1. Приклад роботи консольного додатка для верифікації точності розпізнавання виразів

7.3. Класифікація та сегментація символів

Головною метою цієї секції є оцінка ефективності першого етапу розпізнавання МВ, а саме класифікації та сегментації символів за допомогою НМ. Цей етап також включає попередню обробку вхідних штрихів (згладжування даних, нормалізація, передискретизація тощо).

Для оцінки якості роботи буде використовуватися інтегрована метрика точності розпізнавання виразу (ТРВ). Висока точність розпізнавання символів не завжди гарантує високу якість розпізнавання виразів. Це пов'язано з тим, що на наступному етапі проводиться побудова виразу з огляду на кілька варіантів класифікації символів. Оцінка ефективності методу класифікації і сегментації символів здійснена за допомогою двох експериментів. Під час першого експерименту проведена оцінка запропонованого рішення без додаткової НМ для класифікації окремих символів. Результати представлені у Таблиці 7.2.

Таблиця 7.2. Точність розпізнавання виразів і час класифікації та сегментації символів без додаткового класифікатора окремих символів

	Expression Rate				Середній
	Correct (%)	Top-2 (%)	Top-3 (%)	Top-4 (%)	час (msec)
CROHME 2014	68.8391	76.6802	78.7169	80.3462	12.45
CROHME 2016	66.5214	74.0192	76.2859	77.1578	11.83
CROHME 2019	68.4737	76.7306	78.3153	79.6497	12.31

Під час другого експерименту оцінювався вплив додаткової НМ на точність розпізнавання виразів та швидкість роботи сегментації і класифікації символів. Результати другого експерименту надані у Таблиці 7.3. Як бачимо, додаткова нейронна мережа з невеликою кількістю ваг дає можливість збільшити точність розпізнавання виразів на $3 - 4\%$ (або зменшити помилку розпізнавання виразів на $\approx 10\%$). Але при цьому швидкість сегментації і класифікації зменшується на $\approx 15\%$.

Така зміна швидкості є повністю допустимою як для хмарних рішень, так і для мобільних, і в більшості випадків є непомітною для користувача. А зменшен-

Таблиця 7.3. Точність розпізнавання виразів і час класифікації та сегментації символів з додатковим класифікатором окремих символів

	Expression Rate				Середній
	Correct (%)	Top-2 (%)	Top-3 (%)	Top-4 (%)	час (msec)
CROHME 2014	72.0978	79.5316	81.1609	82.0774	14.58
CROHME 2016	70.5318	77.0706	79.3374	80.0349	14.16
CROHME 2019	71.5596	78.3153	79.8165	80.8173	14.27

ня кількості помилок під час розпізнавання позитивно позначається на зручності роботи з системою.

7.4. Структурний аналіз математичного виразу

7.4.1. Оцінка запропонованих методів мінімізації обчислювальної складності

Завдання цієї секції – це перевірка запропонованих методів для зменшення обчислювальної складності під час структурного аналізу. Експеримент включає почергове включення/виключення кожного елементу, а також усіх їх комбінацій. В цей експеримент включені такі елементи, що мають забезпечувати підвищення швидкодії: променевий пошук, домінуючі символи та регіон охоплення гіпотези. Регіон пошуку не включався в цей експеримент тому, що його вилучення з роботи алгоритму призведе майже до повного перебору усіх комбінацій. Вплив рішення про побудову імен функцій на початку також не вивчався в цьому експерименті, оскільки це призведе до необхідності побудови нових правил виводу граматики, отримання ймовірностей кожного правила та навчання вагових коефіцієнтів впливу моделей. Це, в свою чергу, не дасть можливості порівнювати точність розпізнавання. Інакше кажучи, метод абляцій не може бути застосовний до цього рішення.

Також для цього експерименту в алгоритм структурного аналізу вбудований механізм переривання. Тобто якщо кількість гіпотез на одному рівні стає понад 15 000, процес побудови МВ припиняється. В іншому разі час виконання може перевищувати розумні межі і досягати кількох годин для окремих прикладів.

Всі комбінації були перевірені на трьох наборах даних: CROHME 2014, 2016, 2019. Результати тестів представлені в Таблицях 7.4, 7.5, 7.6 відповідно.

Під час виконання тестів з відключеним променевим пошуком багато прикладів не було розпізнано через вбудовану перевірку на обмеження часу виконання. Для таких прикладів результат розпізнавання містив пустий вираз. Як бачимо, кожен із методів суттєво покращує швидкодію. Але комбінація усіх цих методів допомагає досягти поліпшення швидкості розпізнавання до прийнятних показників. Так, середній час алгоритму розбору зменшився з $6 - 7$ секунд до $20 - 25$ мілісекунд. Також слід зазначити, що інколи застосування регіону охоплення може призвести до незначного погіршення якості розпізнавання. Основна причина погіршення точності розпізнавання пов'язана з наявністю зайвих штрихів, які були сегментовані і класифіковані як окремі символи. Друга причина пов'язана з помилками сегментації, коли один рукописний символ розпізнано як два окремих. У цьому випадку не вдається побудувати вираз, який відповідає правилам граматики.

Алгоритм розбору має складність, яка залежить від кількості символів. Залежність часу розбору виразу від довжини виразу зображено на Рисунку 7.2. Загалом, час розбору залежить від багатьох чинників, таких як кількість варіантів класифікації кожного символу, структура виразу, кількість варіантів просторових відношень тощо.

Очевидно, що точність розпізнавання буде також зменшуватися зі збільшенням довжини виразу (Рисунок 7.3). Це пов'язано з дуже чутливою метрикою ER, яка реагує на будь-яку помилку. Так, для відносно коротких виразів (кількість символів $\leqslant 10$) точність перевищує 75%. Також не відбувається очікуваного різкого зменшення точності відразу в міру збільшення кількості символів у виразі. Це пов'язано з тим, що для коротких виразів часто не вистачає локального контексту для правильної класифікації символів. У тестувальних наборах даних середня довжина виразу складає близько 11 символів. Кількість виразів, які мають довжину понад 30 символів, дуже мала. Для прикладу, кількість зразків з довжиною

Таблиця 7.4. Оцінка ефективності методів мінімізації обчислювальної складності на CROHME 2014

Променевий пошук	Домінуючі символи	Регіон охоплення	ER (%)	SR (%)	Structure Avg. Time (msec)
–	–	–	71.4868	83.2994	6793.28
✓	–	–	71.9959	85.1324	2033.60
–	✓	–	71.5886	83.4012	4063.14
–	–	✓	71.4868	83.4012	4084.52
✓	✓	–	**72.0978**	**85.2342**	480.65
–	✓	✓	71.7923	83.7067	2930.75
✓	–	✓	71.9959	85.0305	758.656
✓	✓	✓	**72.0978**	85.1324	**21.38**

Таблиця 7.5. Оцінка ефективності методів мінімізації обчислювальної складності на CROHME 2016

Променевий пошук	Домінуючі символи	Регіон охоплення	ER (%)	SR (%)	Structure Avg. Time (msec)
–	–	–	68.7881	80.3836	6457.72
✓	–	–	70.3575	83.9582	1501.31
–	✓	–	68.8753	80.7323	5768.09
–	–	✓	68.7881	80.7323	5030.51
✓	✓	–	70.3575	83.9582	451.61
–	✓	✓	68.9625	81.2554	3987.79
✓	–	✓	70.4446	84.0453	107.24
✓	✓	✓	**70.5318**	**84.1325**	**23.56**

Таблиця 7.6. Оцінка ефективності методів мінімізації обчислювальної складності на CROHME 2019

Променевий пошук	Домінуючі символи	Регіон охоплення	ER (%)	SR (%)	Structure Avg. Time (msec)
–	–	–	70.3920	83.0692	6416.18
✓	–	–	71.5596	85.9883	790.66
–	✓	–	70.7256	83.5696	4647.21
–	–	✓	70.5588	83.4028	4095.08
✓	✓	–	**71.6430**	**86.1551**	255.21
–	✓	✓	70.8090	83.9033	3062.55
✓	–	✓	71.4762	85.9049	57.55
✓	✓	✓	71.5596	86.0717	**21.68**

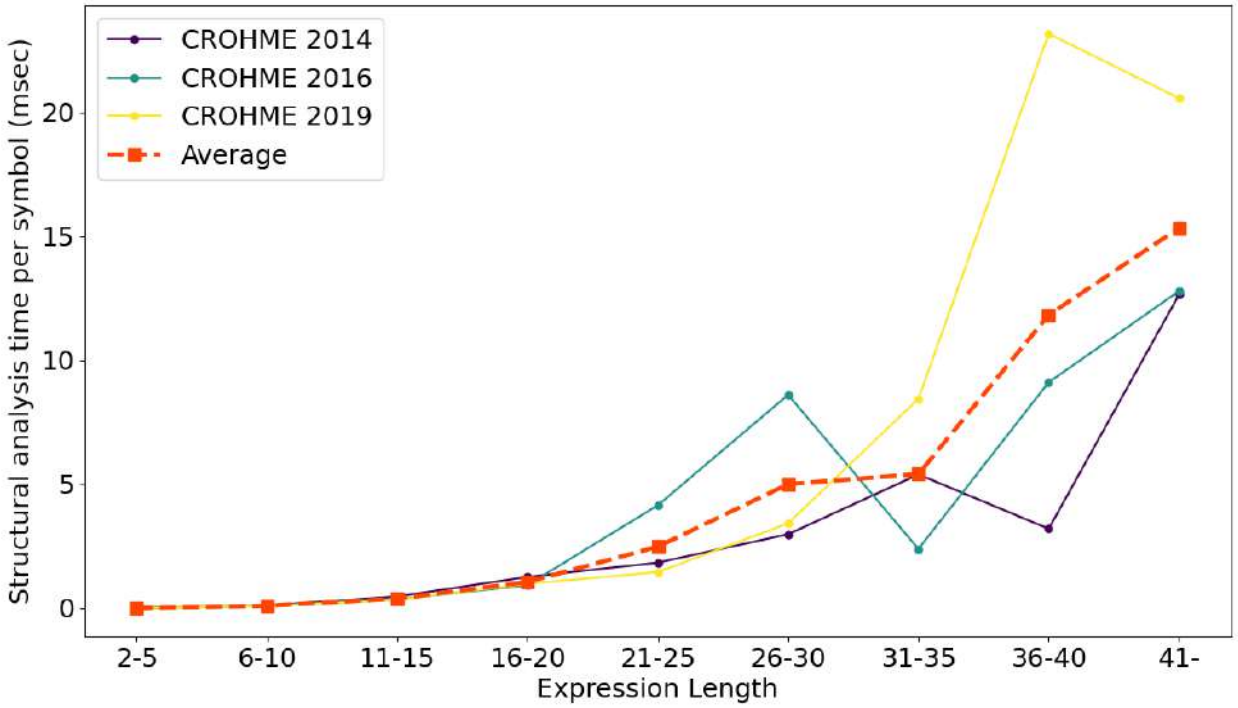


Рис. 7.2. Середній час аналізу структури виразу на один символ залежно від довжини виразу

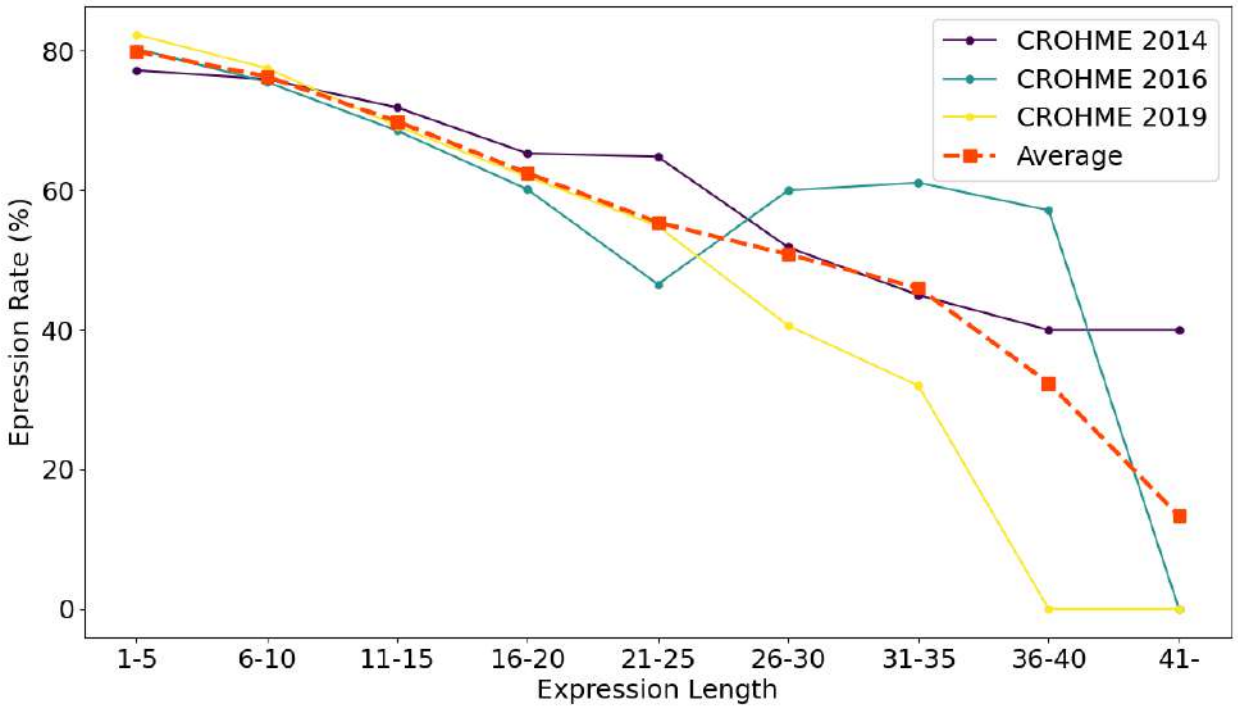


Рис. 7.3. Точність розпізнавання виразу залежно від кількості символів у виразі

понад 40 символів не перевищують 5 в кожному наборі даних. Це призводить до великої похибки у вимірюванні ER для таких зразків.

7.4.2. Мовна модель, ймовірності правил виводу та оцінка впливу вагових коефіцієнтів моделей

Подібно до оцінки методів мінімізації обчислювальної складності в цій секції розглядається вплив на точність розпізнавання мовної моделі та ймовірностей правил виводу граматики. Алгоритмічна складність цих елементів дуже низька, і різниця в часі обробки виразу не перевищує точність вимірювання часу. Тому в цьому розділі будуть розглядатися тільки параметри, пов'язані з точністю. Оцінка впливу здійснювалася шляхом вилучення з формули оцінки ймовірності гіпотези 5.16 елементів, що стосуються відповідної моделі. Так, для видалення мовної моделі встановлювалися нульові значення для відповідних вагових коефіцієнтів ($K^B_{lang} = 0, K^r_{lang} = 0$). Так само для оцінки впливу ймовірностей правил виводу закріплялися нульові значення для K_{bin}, K_{term}. Результати експериментів представлено в Таблиці 7.7.

Таблиця 7.7. Оцінка ефективності моделі мови та ймовірностей правил виводу на CROHME 2014, CROHME 2016, CROHME 2019

Мовна модель	Ймовірності правил виводу	CROHME 2014		CROHME 2016		CROHME 2019	
		ER(%)	SR(%)	ER(%)	SR(%)	ER(%)	SR(%)
–	–	66.19	80.45	65.39	80.30	66.06	81.57
✓	–	70.16	84.11	67.83	82.91	69.06	84.15
–	✓	69.75	83.30	68.53	83.70	69.31	84.99
✓	✓	**72.10**	**85.13**	**70.53**	**84.13**	**71.56**	**86.07**

Застосування мовної моделі та ймовірностей правил виводу дає можливість підвищити точність розпізнавання РМВ на $4 - 6\%$. Для більш кращого розуміння вкладу кожної з моделей зроблено оцінку точності розпізнавання з різними коефіцієнтами. Рисунок 7.4 показує залежність розпізнавання для різних значень коефіцієнтів K^B_{lang}, K_{rel}. Значення інших коефіцієнтів зафіксовані.

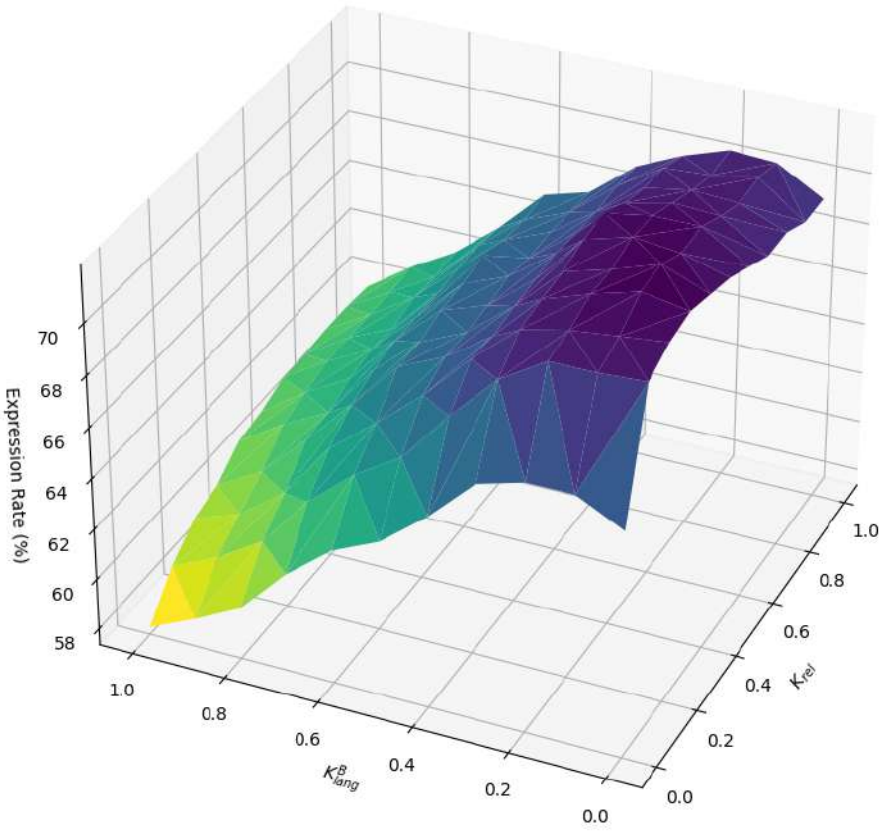


Рис. 7.4. Приклад залежності точності розпізнавання виразів від значення вагових коефіцієнтів моделей для $K_{char} = 0.5$, $K_{term} = 0.02$, $K_{bin} = 0.12$, $K^r_{lang} = 0.07$

7.5. Оцінка швидкості роботи на мобільному пристрої

Для остаточної оцінки швидкості розпізнавання тести запускалися на мобільному пристрої. Для аналізу окремо вимірювалися час сегментації та класифікації символів та час побудови виразу. Агреговані результати аналізу швидкості роботи на мобільному пристрої представлені в Таблиці 7.8. Середній час розпізнавання МВ не перевищує 60 мілісекунд. Також у середньому час, необхідний для структурного аналізу виразу, менший, ніж час на сегментацію та класифікацію символів за допомогою НМ. Але для виразів з великою кількістю вхідних символів час, необхідний для структурного аналізу, може у понад 10 разів перевищувати час класифікації символів. Це пов’язано з тим, що алгоритм структурного аналізу має кубічну часову складність. А алгоритм сегментації і класифікації символів має лінійну складність, яка залежить від кількості вхідних точок. Максимальний

час досягався у розпізнаванні МВ з кількістю символів $\geqslant 41$. Але написання такого виразу на мобільному пристрої ускладнюється обмеженнями розміру екрана. Також система потребує незначних ресурсів у постійній пам'яті ($\approx 4.2MB$), що також спрощує її застосування на мобільних пристроях.

Таблиця 7.8. Параметри швидкодії системи на мобільному пристрої

	CROHME		
	2014	2016	2019
Середній час сегментації та класифікації (msec)	36.63	31.70	32.90
Середній час структурного аналізу (msec)	19.61	18.81	18.58
Середній час розпізнавання виразу (msec)	56.24	50.51	51.48
Мак. час сегментації та класифікації (msec)	244	186	164
Мак. час структурного аналізу (msec)	3245	2354	2805
Середня довжина виразу	10.80	11.18	11.03
Максимальна довжина виразу	99	49	47
Розмір моделей (MB)		2.5	
Бінарний розмір бібліотеки для ARM64 (MB)		1.7	

Розпізнавання виразів з великою кількістю символів завжди буде пов'язане з наявністю помилок. У разі потреби вводу довгих виразів необхідний режим редагування, який дає можливість швидко виправляти помилки розпізнавання. При цьому редагування також здійснюється через рукописні символи та жести. Такий режим забезпечує безшовний та швидкий ввід і редагування МВ [199, 201]. На сьогодні наскрізні рішення не підтримують ітеративний ввід [200]. Цей спосіб розпізнавання дозволяє реалізувати ітеративний метод редагування МВ на основі рукописних жестів.

7.6. Порівняння з наявними методами

До цього ми тестували рішення на моделях, які підтримують 176 математичних символів. У відкритих наборах даних CROHME розмір алфавіту дорівнює 101 символ. Тому для забезпечення порівняння була розроблена модель граматики для підтримки тільки 101 символу. Для символів, які відсутні в CROHME,

була побудована мапа відповідності за подібністю написання. Приклади відповідності: $K \rightarrow k$, $S \rightarrow s$, $Z \rightarrow z$, $\omega \rightarrow w$, $\rho \rightarrow p$, $\eta \rightarrow n$. Таблиця 7.9 містить порівняння точності двох моделей з різним розміром алфавіту. Як і очікувалося, точність розпізнавання збільшується зі зменшенням розміру алфавіту.

Таблиця 7.9. Порівняння точності розпізнавання виразів моделей для розмірів алфавіту 176 та 101 символів

	176 символів		101 символ	
	ER (%)	SR (%)	ER (%)	SR (%)
CROHME 2014	72.0978	85.1324	73.1161	85.3361
CROHME 2016	70.5318	84.1325	71.7524	84.4813
CROHME 2019	71.5596	86.0717	72.3103	86.2385

Перше порівняння, представлене в Таблиці 7.10, містить порівняльний аналіз на підставі даних наукових статей за останні роки. Особливість даного порівняння полягає в тому, що представлені дані найчастіше стосуються наскрізних рішень. Автори зазвичай в описі запропонованого ними методу спираються тільки на відкриті набори даних CROHME для навчання моделей. Розробка цієї системи здійснювалася лише на закритих наборах даних для навчання та верифікації моделей. Як бачимо, точність розпізнавання цієї системи значно перевищує точність у сучасних схожих системах.

Друге порівняння, представлене у Таблиці 7.11, побудоване за результатами відкритих змагань CROHME. Під час проведення цього змагання взагалі не враховується швидкість розпізнавання. Тому, на відміну від результатів, представлених у наукових статтях, часто у змаганнях використовуються різні способи для збільшення точності. Так, можуть застосовувати НМ зі значно більшою кількістю параметрів, складні нейронні моделі мови, ансамблі з багатьох моделей та інші. Модифікованій варіант цієї системи також брав участь в останньому змаганні. Більш детальна інформація подана у Додатку А.1.

Таблиця 7.10. Порівняння точності розпізнавання з наявними підходами на основі публікацій

	CROHME					
	2014		2016		2019	
	ER(%)	SR(%)	ER(%)	SR(%)	ER(%)	SR(%)
BLSTM [196]	30.57	—	—	—	—	—
WAP	46.55	—	44.55	—	—	—
TAP+WAP+LM [192]	61.16	—	57.02	—	—	—
PAL [180]	47.06	—	—	—	—	—
Enhanced MAN [172]	52.43	—	50.86	—	—	—
SBP Attention [179]	54.26	—	51.75	—	—	—
OnSCAN [171]	51.22	70.49	46.12	65.30	46.46	66.31
OnSCAN+TAP [171]	52.64	70.89	47.17	66.78	47.62	67.22
BTTR [198]	53.96	71.40	52.31	69.40	52.96	70.06
Цей метод (176 символів)	72.10	85.13	70.53	84.13	71.56	86.07
Цей метод (101 символ)	**73.12**	**85.34**	**71.75**	**84.48**	**72.31**	**86.24**

Таблиця 7.11. Порівняння точності розпізнавання за результатами змагань

	CROHME					
	2014		2016		2019	
	ER(%)	SR(%)	ER(%)	SR(%)	ER(%)	SR(%)
RIT	18.97	—	—	—	—	—
Nantes	26.06	—	13.34	21.45	—	—
Politècnica de València	37.22	—	—	—	—	—
São Paulo	15.01	—	33.39	57.02	—	—
Tokyo	25.66	—	43.94	61.55	—	—
Wiris	—	—	49.61	74.28	—	—
USTC-iFLYTEK	—	—	—	—	**80.73**	**91.49**
Samsung R&D 1	—	—	—	—	79.82	89.32
MyScript	62.68	—	67.65	**88.14**	79.15	90.66
Sun Yat-Sen U.	—	—	—	—	77.40	88.82
Samsung R&D 2	—	—	—	—	65.97	82.82
PAL-v2	—	—	—	—	62.55	79.15
MathType	—	—	—	—	60.13	79.15
TUAT	—	—	—	—	39.95	58.22
Цей метод (176 символів)	72.10	85.13	70.53	84.13	71.56	86.07
Цей метод (101 символ)	**73.12**	**85.34**	**71.75**	84.48	72.31	86.24

7.7. Висновки до Розділу 7

У розділі проведений детальний аналіз запропонованого методу розпізнавання РМВ. Оскільки процес розпізнавання складається з двох головних етапів, то кожен етап розглядався окремо, але враховуючи остаточний результат. Так, були проаналізовані та представлені метрики для оцінювання точності розпізнавання; побудована серія експериментів із застосуванням методу абляцій для вивчення впливу окремих елементів на якість та швидкість розпізнавання. Усі експерименти побудовані на відкритих наборах даних CROHME.

Запропонований метод для підвищення точності сегментації та класифікації окремих символів дав можливість підвищити точність розпізнавання МВ на $2-3\%$. Була проведена оцінка методів оптимізації структурного аналізу. В результаті їх застосування середній час структурного аналізу скоротився з $5-7$ секунд до $20-40$ мілісекунд. Це допомогло досягти необхідних значень часу розпізнавання, що пред'являються до обчислень в рамках обмеження ресурсів. Так, середній час розпізнавання на мобільному пристрої складає близько 55 мілісекунд. Також це рішення має невисокі вимоги до розміру постійної пам'яті та забезпечує можливість побудови рішення з редагування в інтерактивному режимі.

Проведено порівняння з наявними методами, спираючись на відкриті джерела інформації. Для можливості порівняння була розроблена модель, яка підтримує 101 символ. Рішення показує значно кращі показники точності розпізнавання порівняно з рішеннями, представленими в наукових статтях.

РОЗДІЛ 8

ПРАКТИЧНЕ ЗАСТОСУВАННЯ ТА ІНТЕЛЕКТУАЛЬНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА

Досягнення в галузі систем глибокого навчання для задач розпізнавання дають можливість розвивати інтелектуальний інтерфейс користувача. Водночас вони призводять до нових вимог та обмежень у його дизайні. Оскільки інтерфейс вводу за допомогою пера є більш придатним для 2D мов з точки зору взаємодії людини з комп'ютером, особливий інтерес дослідницької спільноти був приділений розробці інтелектуальних інтерфейсів користувача для роботи з математичними виразами [130, 161].

У цьому розділі наводяться приклади застосування розробленої системи розпізнавання МВ на мобільних пристроях: рукописний калькулятор, «Інтерактивний папір» та «S Note». Особлива увага приділяється побудові інтелектуального інтерфейсу користувача. Оцінка зручності була досліджена шляхом опитування користувачів та порівняння швидкості вводу різними способами. Рукописний калькулятор та «Інтерактивний папір» – це протопити додатків, які розроблені всередині компанії *Samsung Research and Development Institute Ukraine (SRK)*. Головною метою розробки цих додатків є демонстрація можливостей технологій рукописного вводу та дослідження інтерфейсів користувача на основі пера. Практичного застосування запропонована система набула у додатку «S Note».

Необхідно відзначити, що автор брав участь у дослідженнях зручності, розробці концепції інтерфейсу для рукописного калькулятора та додатка «Інтерактивний папір», а також в інтеграції розробленої системи розпізнавання МВ у всі додатки, але розробка самих додатків не є досягненням автора цієї роботи.

8.1. Прототип рукописного калькулятора для вводу, редагування та обчислення математичних виразів

В останні десятиліття були розроблені різні програми для рукописних калькуляторів [153, 157, 158, 189]. Так, для демонстрації можливостей розробленої системи розпізнавання МВ був розроблений мобільний додаток з урахуванням таких припущень:

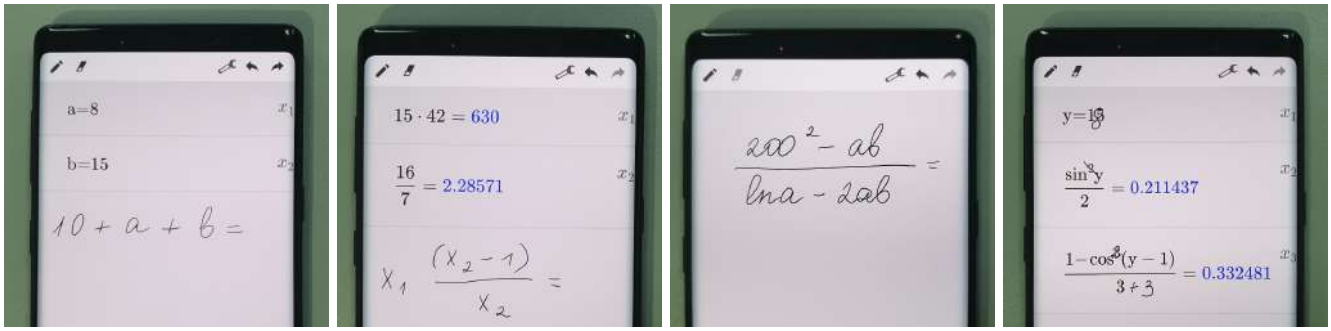
- функція **редагування** є невід'ємною частиною будь-якого додатка під час вводу інформації;
- підтримка **визначених користувачем змінних** може значно скоротити зусилля користувачів та необхідний для виконання розрахунків час. Зіткнувшись з необхідністю перерахувати один і той самий вираз з різними значеннями або ввести довгий складний вираз на маленькому екрані, користувачі, можливо, віддадуть перевагу вводу змінних під час вводу МВ;
- **автоматичні змінні** (автоматично призначається ім'я змінної для кожного виразу/екземпляра) можуть значно спростити повторне використання результатів попередніх виразів. Більшість наявних калькуляторів надають доступ до історії, а автоматичні змінні пропонують більш зручний і гнучкий спосіб доступу до результатів попередніх обчислень.

Наведені вище припущення щодо функціональності мають покращити взаємодію з додатком. Проте наявні програми слабо підтримують ці функції. У більшості випадків підтримка змінних доступна тільки в професійних пакетах, а безкоштовні версії калькуляторів не підтримують роботу зі змінними і дають доступ лише до історії розрахунків. Багато програм не підтримують редагування МВ за допомогою рукописних жестів. Також не існує наборів даних, спеціально розроблених для вивчення зручності вводу та оцінки точності редагування. Таблиця 8.1 демонструє порівняння між кількома різними системами з точки зору запропонованих вимог до інтерфейсу користувача.

На Рисунку 8.1 наведено дизайн додатка та приклади рукописного вводу МВ для обчислення.

Таблиця 8.1. Порівняння функціональності додатка рукописного калькулятора з попередніми роботами з точки зору запропонованих вимог до інтерфейсу користувача

Робота	Змінні користувача	Автоматичні змінні	Редагування
PenCalc [27]			
MathPad^2 [93]	✓		✓
Zeleznik et al. [189]	✓		✓
MathBrush [114]			✓
Microsoft Math Solver [117]	✓		
MyScript Calculator [125]			✓
Запропонований додаток [199]	✓	✓	✓



(а) Приклад змінних користувача (б) Приклад автоматичних змінних (в) Приклад рукописного вводу зі злитим написанням (г) Приклад рукописного редагування

Рис. 8.1. Функції рукописного калькулятора: рукописний ввід, розпізнавання, обчислення, редагування, змінні

Основні функціональні частини системи показані на Рисунку 8.2. Модуль кластеризації (*clusterization*) розбиває набір штрихів на групи. Кожна група розпізнається окремо і може включати штрихи для кількох символів. Сегментація та класифікація символів, побудова виразу виконані на базі методів, описаних у попередніх розділах. Модуль розрахунків (*solver*) підтримує основні математичні операції і обчислює вирази. Модуль візуалізації (*renderer*) отримує математичні вирази в нотації LaTeX, визначає розташування та відображення кожного символу. Модуль контролера редагування (*editor controller*) відповідає за загальний контроль ітераційного процесу вводу, керування екземплярами та забезпечує простий програмний інтерфейс.

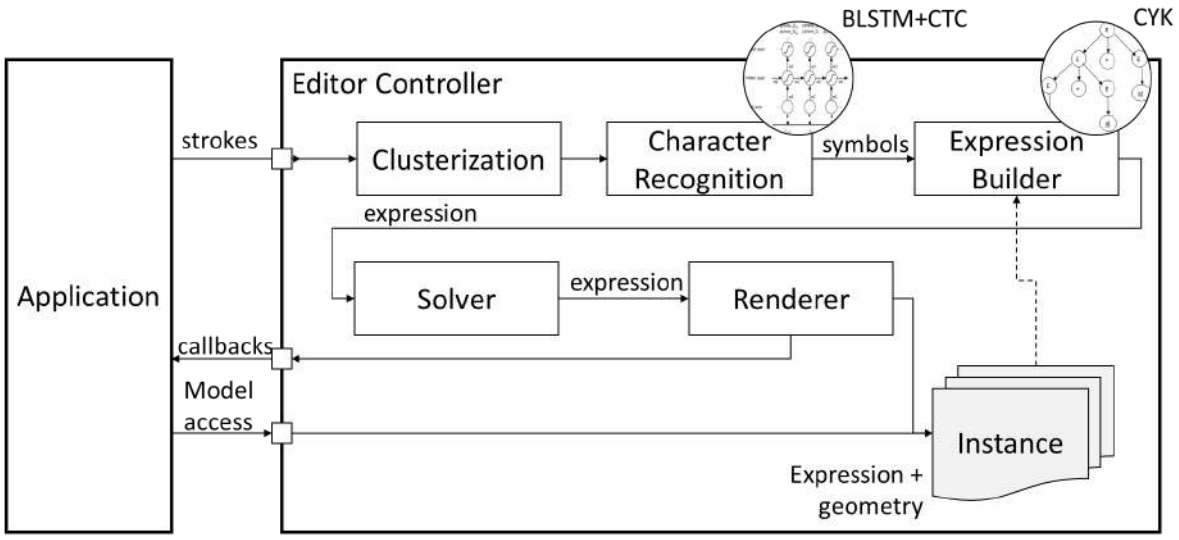


Рис. 8.2. Загальна архітектура рукописного калькулятора

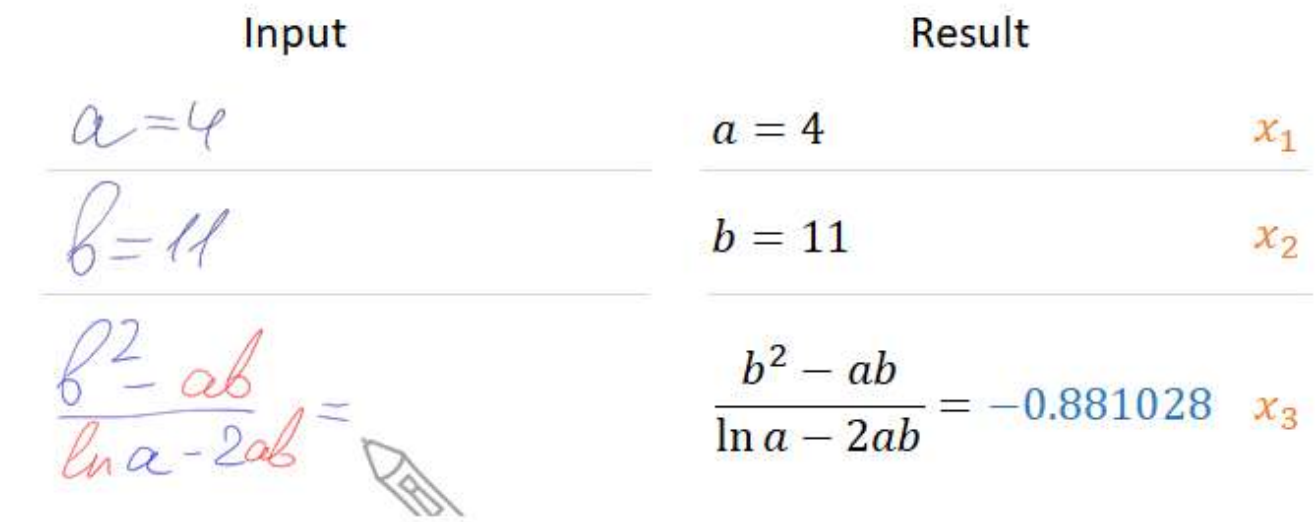
8.1.1. Інтерфейс користувача

Інтерфейс користувача являє собою єдине полотно для вводу МВ. Він покликаний нагадувати написання на папері, яке є природним і знайомим кожному користувачеві з початкової школи. Усі вхідні дані перетворюються на друковане представлення, яке дає можливість користувачеві бачити, що обчислюється. Такий підхід допомагає усунути невизначеність результату розрахунку для користувача.

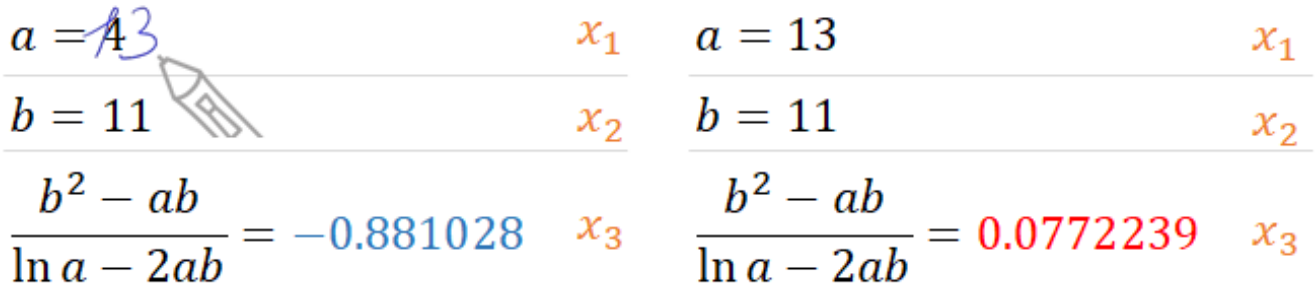
Користувачі можуть працювати з кількома екземплярами математичних виразів одночасно. Одночасний доступ до багатьох екземплярів знижує складність вхідних виразів, зменшує навантаження на пам’ять користувача, спрощує структуру та процес вводу. Для кожного екземпляра МВ вводиться іменована змінна, яку можна використовувати в наступних обчисленнях. Рисунок 8.3 містить приклади різних сценаріїв.

8.1.2. Зручність інтерфейсу

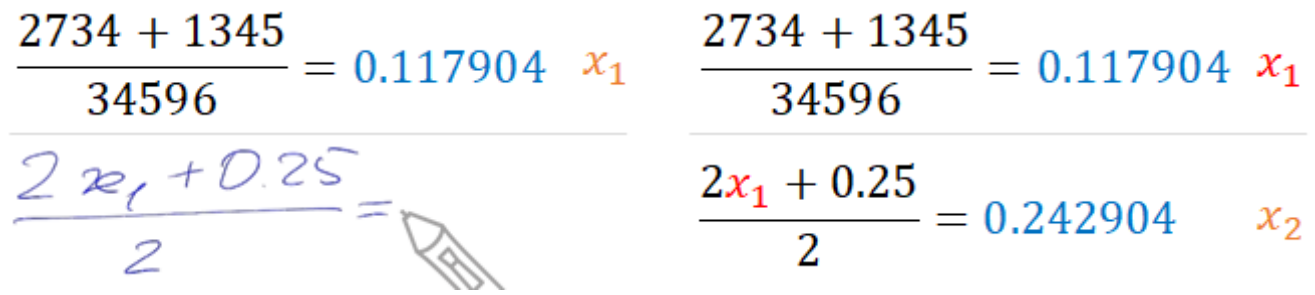
Оцінка зручності запропонованого інтерфейсу рукописного калькулятора була проведена за допомогою серії експериментів, які включали також оцінку додаткових запропонованих функцій. Більш детальну інформацію про дослідження можна знайти у роботі [199]. В цій секції містяться основні відомості про експеримент та висновки щодо зручності рукописного калькулятора.



(а) Ввід багатьох екземплярів та зв’язне написання



(б) Редагування та явні специфіковані змінні



(в) Робота з автоматичними змінними

Рис. 8.3. Приклади взаємодії користувача в рукописному калькуляторі

У дослідженні взяли участь 30 осіб. Для учасників було підготовлено п’ять завдань. Усі завдання може розв’язати кожен, кому виповнилося 14 років, адже завдання базуються на шкільній програмі. Приклади завдань зображені на Рисунку 8.4.

Після експерименту більшість користувачів відповіли, що віддадуть перевагу рукописному вводу над традиційним (за допомогою віртуальної чи фізичної клавіатури) для рішення математичних завдань. 18.1% користувачів віддали пе-

1. Find result:

$$\frac{b^2 - ab}{\ln a - 2ab} =$$

a) a = 4, b = 11
b) a = 13, b = 25

2. Check equality for given values (calculate both sides and compare):

$$\sin^2 \frac{xy}{2} = \frac{1 - \cos xy}{2}$$

a) x = 12, y = 7

3. Find out how much will be paid to you if you put $ 500 :

a) 3%, 4 years
b) 4%, 3 years

$$x = (1 + p)^n - s$$

where:
s – deposit, p – percent,
n – term, x – income

4. Choose the bank for currency exchange:

Bank A: exchange rate = 24.12
no commission
Bank B: exchange rate = 24.17
commission = 10

a) $150
b) $200

5. Yesterday you have ordered several pizzas and paid X$. You remember ordering cheese pizza and pepperoni pizza, but you don't remember how many of each you ended up buying. Cheese pizza cost A$, and pepperoni cost B$. Try to find out how many pizzas of each type you ordered.

Рис. 8.4. Приклади завдань для дослідження зручності запропонованого інтерфейсу рукописного калькулятора

ревагу традиційному калькулятору, а 36.4% відповіли, що для них немає різниці між традиційним та рукописним калькуляторами. Користувачі, які раніше не мали досвіду рукописного інтерфейсу, були настільки вражені, що 77.8% віддали перевагу рукописному калькулятору.

Результат експерименту показав, що користувачі активно використовують редагування як для виправлення помилок розпізнавання, так і для внесення змін у вихідні дані. Також калькулятор, що підтримує імена змінних, був позитивно сприйнятий користувачами та допомагає скоротити час, необхідний для вирішення багатьох типів завдань. А підтримка декількох екземплярів дає можливість нівелювати обмеження розміру екранів мобільних пристроїв.

8.2. «Інтерактивний папір» та документи зі складною структурою

Рукописний калькулятор є прикладом додатка з мінімалістичним дизайном інтерфейсу на основі пера. Але такий додаток є досить вузькоспеціалізованим. Часто користувачеві необхідно створювати документи, які одночасно можуть містити різні елементи, такі як текст, таблиці, графіки, діаграми та МВ. Підтримка створення таких документів вимагає інтеграції множини елементів, кожен із яких відповідає за розпізнавання певного типу елементу. Наразі одним із поширених способів є попереднє ручне налаштування областей в документі для кожного типу елементу. Прикладом такого може бути виділення окремого простору на сторінці документа для вводу МВ. У цьому випадку вибір механізму розпізнавання визначається типом області. Але це призводить до збільшення складності інтерфейсу користувача. Тобто користувач спочатку має вказати тип елементу, що вводиться, і визначити його розташування на сторінці.

Методи розпізнавання рукописного тексту вдосконалюються з кожним роком. Незважаючи на успіхи, розпізнавання рукописних елементів є схильним до помилок через різні стилі рукописного вводу та різні навички користувачів [138]. Тому система має передбачати методи швидкого виправлення помилок. Ітераційний процес вводу документу більш зручніший завдяки можливості перевірки та виправлення помилок у разі необхідності.

У роботі [201] була запропонована система «Інтерактивний папір» для ітераційного вводу документів на мобільних пристроях. Ця система за допомогою методів розпізнавання рукописних штрихів забезпечує ввід та редагування документів, що містять різноманітні елементи. Процес вводу елементів документа та результат зображені на Рисунку 8.5.

Запропонована система забезпечує ітераційне розпізнавання об’єктів з урахуванням попереднього вмісту. Рисунок 8.6 ілюструє архітектуру системи з точки зору обробки рукописних жестів. По-перше, система розпізнає спеціальні жести редагування, що пов’язані з простою модифікацією документа. Після цього виконується сегментація та класифікація штрихів [39, 64, 84]. Модуль розпізнавання

(а) Користувач малює елементи документа за допомогою стилуса

(б) Кругова діаграма та таблиця розпізнаються та перетворюються на відповідні елементи, користувач додає рукописний текст

(в) Рукописний текст розпізнається, перетворюється на друкований текст та застосовується до відповідних елементів документа

Рис. 8.5. Приклад ітеративного процесу вводу елементів документа

жестів редагування та модуль класифікації штрихів працюють з поточною моделлю документа, що усуває більшість неоднозначностей у розпізнаванні штрихів. За результатами класифікації штрихів застосовується відповідний модуль розпізнавання. Результатом цього кроку є набір об’єктів, що також містять інформацію про початкову геометрію. На останньому кроці виконується об’єднання нового розпізнаного об’єкта з моделлю документа.

У цьому додатку запропонований метод розпізнавання МВ, інтегрований з алгоритмами розпізнавання інших типів елементів в єдину систему. На діаграмі модуль розпізнавання МВ (*Math recognition*) відповідає лише за сегментацію та класифікацію математичних символів. Структурний аналіз математичного виразу виконується на наступному етапі (*Object merger*). Це пояснюється тим, що у побудові виразу необхідно враховувати не лише нові розпізнані символи, а й попередні.

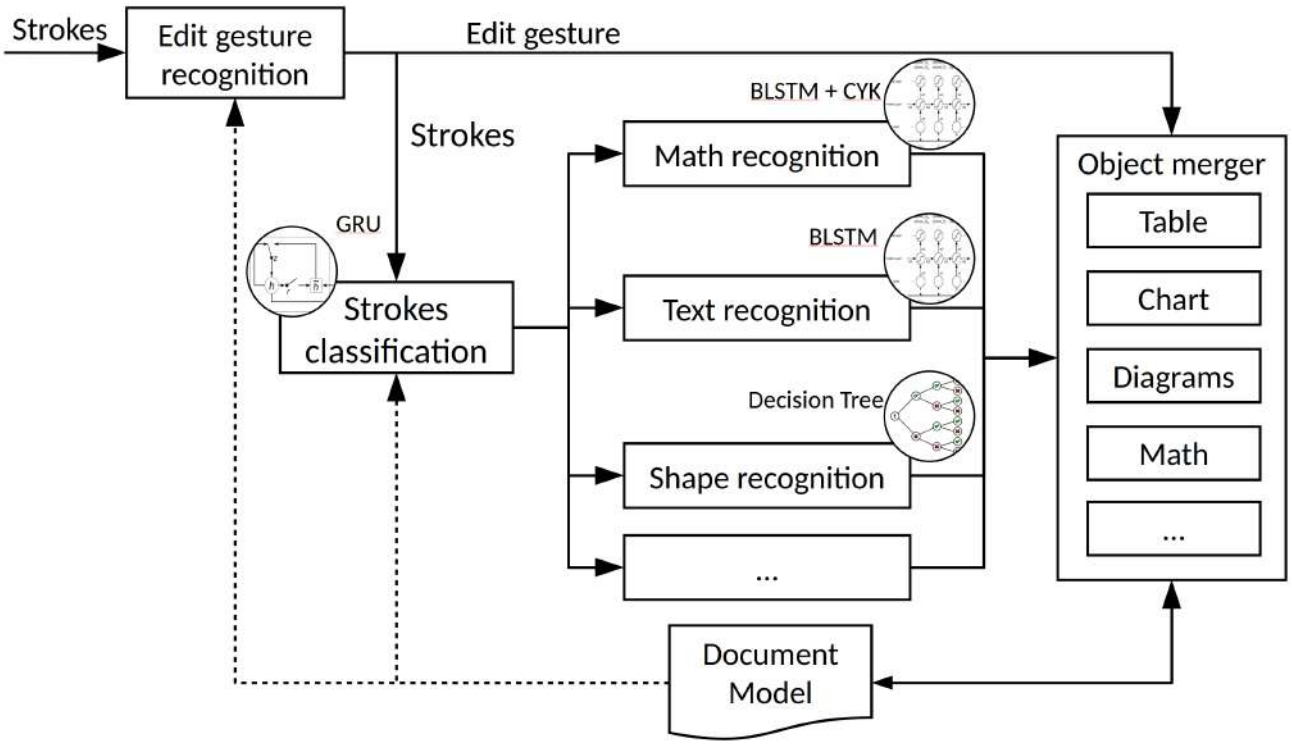


Рис. 8.6. Загальна послідовність ітеративного розпізнавання рукописних елементів документа

Запропонований метод ітераційного вводу забезпечує підтримку багатьох типів елементів документа. Перелік типів елементів та їх приклади наведено на Рисунку 8.7.

Більшість людей можуть друкувати текст швидше, ніж писати [22]. Але таке твердження справедливе лише для простого тексту (1D-мова). Тому оцінка зручності користування здійснювалась шляхом порівняння швидкості створення маленького документа 8.8, що містить різні типи елементів. На мобільному пристрої використовувався додаток «Інтерактивний папір» з інтерфейсом на основі пера. Офісний пакет *Microsoft Office* застосовувався для порівняння з інтерфейсом на базі клавіатури та мишки. В результаті було визначено, що інтерфейс вводу за допомогою пера забезпечує зменшення часу вводу в $1.5 - 2$ рази.

Головне обмеження інтерфейсу на базі пера – це відсутність достатньої кількості інформації для правильної класифікації типу об’єкта [84]. Для прикладу, окремо нарисований круг може виступати як елемент діаграми, текст, малюнок або МВ. Також існують суперечності, пов’язані з перекриттям об’єктів. Прикла-

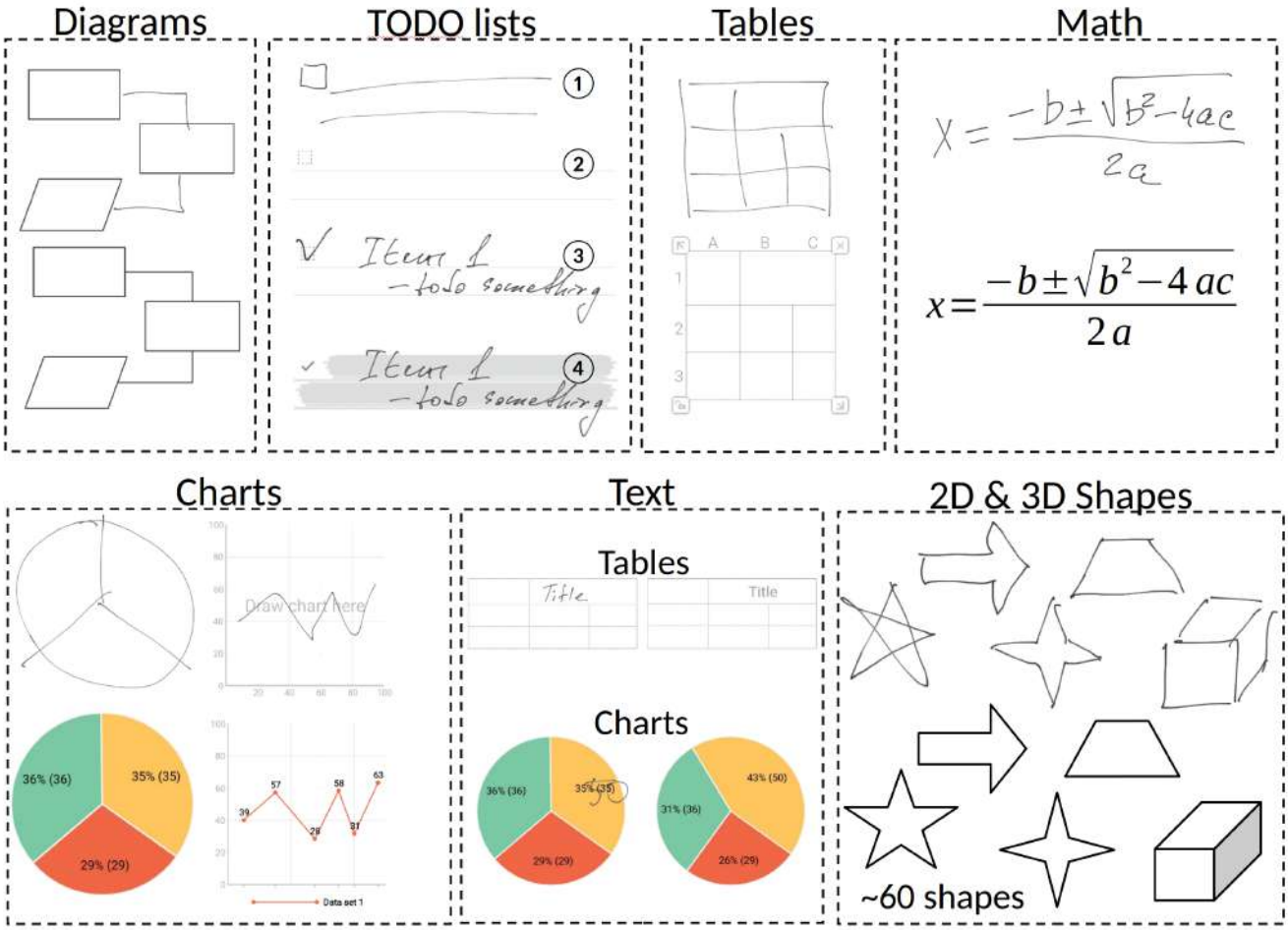


Рис. 8.7. Елементи документа, що підтримуються у додатку «Інтерактивний папір»

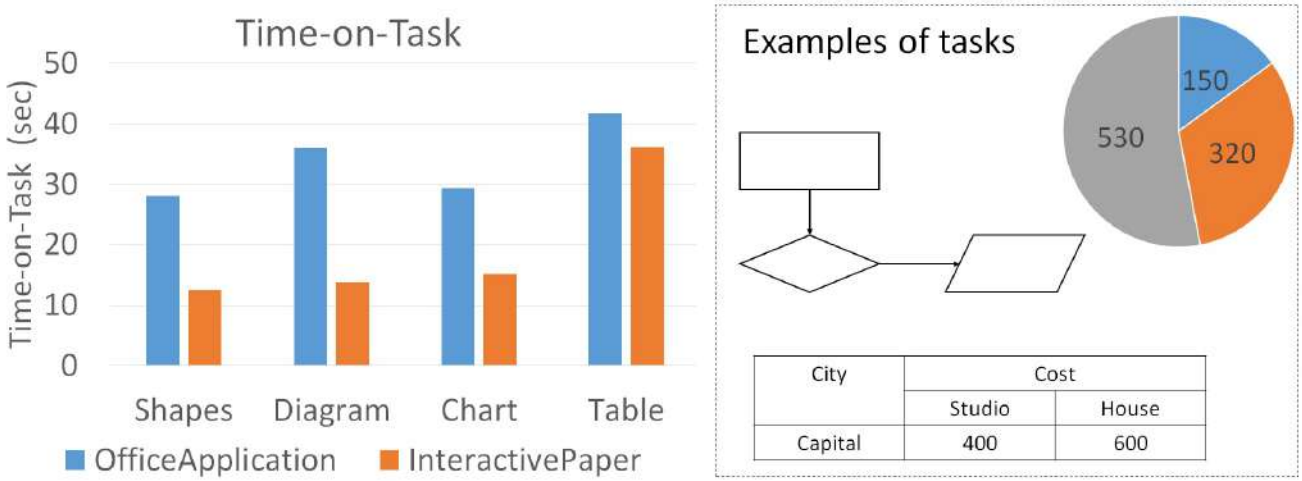
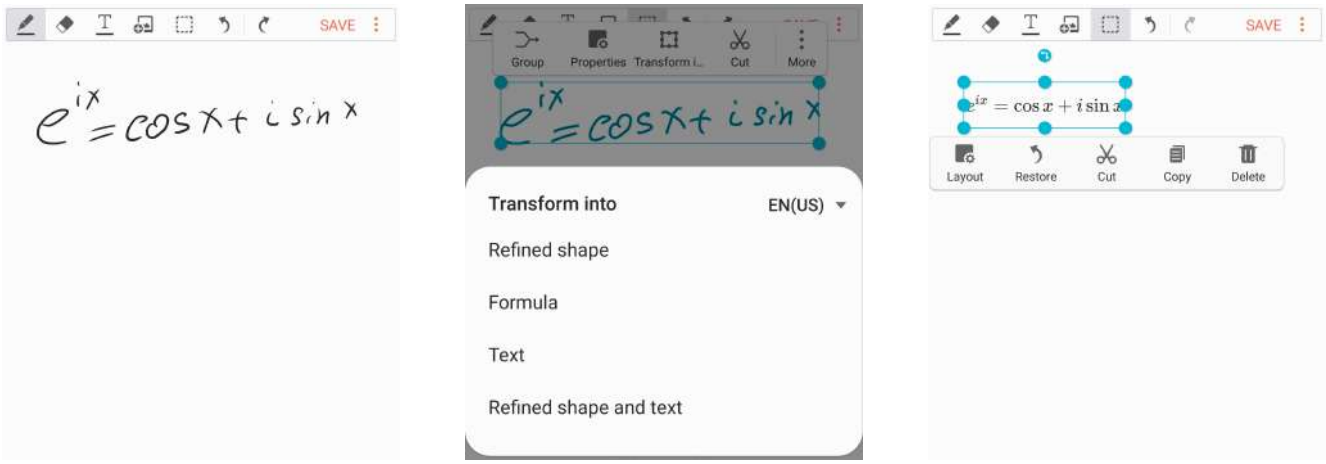


Рис. 8.8. Приклад завдання та результати дослідження зручності інтерфейсу з пером для вводу 2D-елементів документа в додатку «Інтерактивний папір»

дом цього випадку є текстовий блок, який був намальований на кількох клітинках таблиці.

8.3. Додаток «S Note»

Запропонований метод розпізнавання МВ був інтегрований в комерційний додаток «S Note», що поставлявся з мобільними пристроями *Samsung Galaxy Note 9*. Розпізнавання здійснювалося у пакетному режимі, в якому користувач вибирає набір штрихів, які потрібно трансформувати в друковане представлення МВ. Приклад взаємодії користувача з додатком «S Note» зображено на Рисунку 8.9.



(а) Користувач вводить рукописний МВ

(б) Користувач виділяє рукописні штрихи та вибирає елемент меню *Transform* → *Formula*

(в) Замість вхідних штрихів вставляється результат розпізнавання в друкованому вигляді

Рис. 8.9. Приклад інтерфейсу користувача для взаємодії з модулем розпізнавання РМВ в пакетному режимі у додатку «S Note».

8.4. Висновки до Розділу 8

В розділі продемонстровано приклади застосування системи розпізнавання МВ для різноманітних мобільних додатків з інтерфейсом на основі пера. Розглянуті додатки більш зручні та забезпечують швидкий та природний ввід 2D-елементів документів порівняно з інтерфейсами на базі клавіатури та мишки. Для рукописного калькулятора був запропонований інтерфейс, що підтримує ввід багатьох екземплярів МВ та обчислення на основі змінних. Слід зазначити наяв-

ність режиму редагування МВ, що забезпечує можливість ітеративного вводу та виправлення помилок, які могли виникнути у розпізнаванні РМВ. Розроблена система розпізнавання РМВ повною мірою покриває потреби для реалізації простого калькулятора і має низку переваг. Додаток «Інтерактивний папір» надає інтелектуальний інтерфейс для вводу та редагування багатьох типів елементів за допомогою рукописного вводу. Серед підтримуваних типів елементів представлені текст, таблиці, діаграми і, зокрема, МВ.

Представлений у цій роботі метод розпізнавання РМВ впроваджений у комерційний додаток «S Note». Розроблений модуль інтегрований у пакетному режимі у вигляді окремого плагіна для трансформації набору рукописних штрихів у друковане представлення МВ.

ВИСНОВКИ ТА ПОДАЛЬША РОБОТА

У дисертаційній роботі досліджувалась проблема онлайн розпізнавання рукописних математичних виразів. В результаті для вирішення цієї проблеми було запропоновано новий підхід, який складається з двох етапів: 1) сегментація та класифікація символів за допомогою ДДКЧП та НЧК; 2) структурний аналіз на основі контекстно-вільних граматик. Головною особливістю запропонованого методу є орієнтація на пристрої з обмеженими ресурсами, підтримка пакетного та ітеративного режиму розпізнавання. Ефективність методу була перевірена на відкритих наборах даних CROHME та на мобільних пристроях.

Окрім цього, був запропонований метод для автоматичної анотації рукописних прикладів, що суттєво зменшило час на підготовку наборів даних для тренування та тестування. Також були продемонстровані приклади мобільних додатків з інтегрованим рішенням для розпізнавання РМВ та проведена оцінка зручності інтерфейсу користувача на основі пера.

У дисертаційній роботі отримано такі основні результати:

1. Вперше запропонована система для онлайн розпізнавання рукописних математичних виразів, що базується на ДДКЧП та НЧК для одночасного вирішення задач сегментації та класифікації символів для 2D мов, що забезпечує високу точність розпізнавання.
2. Запропонована додаткова легковагова НМ для класифікації окремих символів та вперше розроблено алгоритм для інтеграції двох НМ в ансамбль, що вирішує декілька задач одночасно. Це дало можливість суттєво зменшити похибку розпізнавання при мінімальних додаткових витратах по часу та пам’яті.
3. Покращена концепція «body box» за рахунок більшої деталізації класів символів та локального контексту та запропоновано новий набір з 10 гео-

метричних ознак, який використовує «body box» і обмежувальний прямокутник. Це дозволило підвищити точність класифікації просторових відношень.

4. Покращено функцію для визначення найбільш ймовірного МВ, що забезпечує інтеграцію моделі сегментації та класифікації символів, класифікації просторових відношень, мовної та граматичної моделей у єдиному показнику.
5. Вперше адаптовані та застосовані домінуючі символи для граматичних підходів структурного аналізу. Це дає можливість знизити складність алгоритму розбору МВ з вертикально розташованими елементами.
6. Запропоновано та розроблено набір підходів, що спираються на геометричний пошук та динамічне програмування для зменшення складності алгоритму:
   — променевий пошук з динамічним регулюванням ширини променя;
   — регіони пошуку;
   — регіони охоплення.

   Все це разом допомагає наблизити обчислювальну складність алгоритму структурного аналізу до $O(N^{3+\varepsilon} \cdot \log N \cdot |G(P)|)$, де $|G(P)|)$ – кількість правил виводу в граматиці, а N – кількість символів. При цьому система формує результат, який містить декілька найбільш ймовірних МВ.
7. Вперше був запропонований ітеративний підхід для напівавтоматичної анотації РМВ, що дозволяє значно зменшити час підготовки наборів даних для тренування та тестування. Запропонований метод може застосовуватися для анотації наборів даних з високим рівнем шуму та для виявлення підозрілих зразків та зразків з помилками. Точність анотації виразів за допомогою цього методу перевищує 98.9%.
8. Проведено обчислювальні експерименти на відкритих наборах даних CROHME. Було перевірено ефективність запропонованих методів підвищення якості розпізнавання та зменшення обчислювальної складності. Основні результати експериментів:

— точність розпізнавання при розмірі алфавіту в 101 символ становить 73.12%, 71.75% та 72.31% на CROHME 2014, 2016, 2019 відповідно;
— середній час розпізнавання РМВ на мобільному пристрої становить $50msec$ при середній довжині виразу в ≈ 11 символів;
— додаткова легковагова НМ дала можливість зменшити помилку розпізнавання виразу на $\approx 10\%$, при цьому час класифікації та сегментації символів збільшився на $\approx 15\%$;
— запропоновані методи для зменшення обчислювальної складності алгоритму дозволили скоротити час виконання структурного аналізу з декількох секунд до $17 - 40$ мілісекунд.

9. Продемонстровано приклади застосування системи розпізнавання МВ для побудови різноманітних мобільних додатків з інтерфейсом на основі пера. Були представлені мобільні додатки: 1) рукописний калькулятор для вводу, редагування та обчислення математичних виразів; 2) «Інтерактивний папір» для вводу складних документів; 3) «S Note».
10. Досліджено зручність користування інтерфейсом з пером. Дослідження здійснювалось шляхом виконання однакових завдань у різних додатках відносно великою кількістю учасників. Час виконання деяких завдань за допомогою інтерфейсу на основі пера у $1.5 - 2$ рази менший, ніж при використанні клавіатури та мишки в стандартних додатках.

Представлений у роботі метод розпізнавання РМВ був впроваджений у комерційний додаток «S Note» в пакетному режимі розпізнавання.

Спираючись на аналіз помилок представленої системи розпізнавання, вимоги та сучасні тенденції розвитку систем, сформульовані можливі напрямки подальшої роботи:

1. Підвищення точності розпізнавання можливе за рахунок більш сучасних моделей мови, що будуються на НМ та здатні забезпечити доступ до більш довгострокових залежностей між елементами виразу.
2. Ще одним напрямком, що пов'язаний з підвищенням точності розпізнавання, є персоналізація. Донавчання моделі розпізнавання під конкретного

користувача може суттєво зменшити неоднозначності, що пов’язані з особливостями написання.

3. Матриці поширені в математиці, але розпізнавання матриць вимагає більш складного аналізу структури виразу. Підтримка розпізнавання матричних структур виходить за рамки цієї роботи. Варто відзначити, що запропонований метод дає можливість додати підтримку розпізнавання матриць, і роботи в цьому напрямку вже ведуться [30, 182].
4. Мультимодальний ввід математичних виразів може позитивно вплинути на інтерфейс взаємодії з користувачем. Особливо це може стосуватися режиму редагування або виправлення помилок розпізнавання.

СПИСОК ІЛЮСТРАЦІЙ

СПИСОК ТАБЛИЦЬ

СПИСОК АЛГОРИТМІВ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Frank DJ Aguilar and Nina ST Hirata. 2012. ExpressMatch: a system for creating ground-truthed datasets of online mathematical expressions. In *Proc. Int. Workshop on Document Analysis Systems*. 155–159.

[2] Francisco Alvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. 2011. Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1225–1229.

[3] Francisco Alvaro, Joan-Andreu Sánchez, and Jose-Miguel Benedi. 2012. Unbiased evaluation of handwritten mathematical expression recognition. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 181–186.

[4] Francisco Alvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. 2013. Classification of on-line mathematical symbols with hybrid features and recurrent neural networks. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1012–1016.

[5] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. 2014. Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden Markov models. *Pattern Recognition Letters* 35 (2014), 58–67.

[6] Francisco Alvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. 2016. An integrated grammar-based approach for mathematical expression recognition. *Pattern Recognition* 51 (2016), 135–147.

[7] Francisco Alvaro and Richard Zanibbi. 2013. A shape-based layout descriptor for classifying spatial relationships in handwritten math. In *Proceedings of the 2013 ACM symposium on Document engineering*. 123–126.

[8] Francisco Álvaro Muñoz, Joan Andreu Sánchez Peiró, and José Miguel Benedí Ruiz. 2011. *IMEGE: Image-based Mathematical Expression Global Error*. technical report. Universitat Politécnica de Valéncia.

[9] Walaa Aly, Seiichi Uchida, Akio Fujiyoshi, and Masakazu Suzuki. 2009. Statistical classification of spatial relationships among mathematical symbols. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1350–1354.

[10] Robert H Anderson. 1967. Syntax-directed recognition of hand-printed two-dimensional mathematics. In *Proc. ACM Symposium on Interactive Systems for Experimental Applied Mathematics*. 436–459.

[11] Lisa Anthony, Jie Yang, and Kenneth R Koedinger. 2005. Evaluation of multimodal input for entering mathematical equations on the computer. In *Proc. ACM CHI Extended Abstracts on Human Factors in Computing Systems*. 1184–1187.

[12] Ahmad-Montaser Awal, Harold Mouchere, and Christian Viard-Gaudin. 2009. Towards handwritten mathematical expression recognition. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1046–1050.

[13] Ahmad-Montaser Awal, Harold Mouchère, and Christian Viard-Gaudin. 2010. A hybrid classifier for handwritten mathematical expression recognition. In *Proc. of SPIE*, Vol. 7534. 1–10.

[14] Ahmad-Montaser Awal, Harold Mouchere, and Christian Viard-Gaudin. 2010. The problem of handwritten mathematical expression recognition evaluation. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 646–651.

[15] Ahmad-Montaser Awal, Harold Mouchère, and Christian Viard-Gaudin. 2014. A global learning approach for an online handwritten mathematical expression recognition system. *Pattern Recognition Letters* 35 (2014), 68–77.

[16] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. 2011. Sequential deep learning for human action recognition. In *International workshop on human behavior understanding*. Springer, 29–39.

[17] Zhen-Long Bai and Qiang Huo. 2005. A study on the use of 8-directional features for online handwritten Chinese character recognition. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. IEEE, 262–266.

[18] Stephen Balaban. 2015. Deep learning and face recognition: The state of the art. In *Biometric and Surveillance Technology for Human and Activity Identification XII*, Vol. 9457.

[19] Barbara Beeton, Asmus Freytag, and Murray Sargent III. 2017. *Unicode support for mathematics*. technical report. The Unicode Consortium.

[20] Abdelwaheb Belaid and Jean-Paul Haton. 1984. A syntactic approach for handwritten mathematical formula recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 1 (1984), 105–111.

[21] Xiaohang Bian, Bo Qin, Xiaozhe Xin, Jianwu Li, Xuefeng Su, and Yanfeng Wang. 2021. Handwritten Mathematical Expression Recognition via Attention Aggregation based Bi-directional Mutual Learning. *arXiv preprint arXiv:2112.03603* (2021).

[22] C Marlin "Lin" Brown. 1988. Comparison of typing and handwriting in "two-finger typists". In *Proceedings of the Human Factors Society Annual Meeting*, Vol. 32. SAGE Publications Sage CA: Los Angeles, CA, 381–385.

[23] Mehmet Celik and Berrin Yanikoglu. 2011. Probabilistic mathematical formula recognition using a 2D context-free graph grammar. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 161–166.

[24] Chungkwong Chan. 2020. Stroke extraction for offline handwritten mathematical expression recognition. *IEEE Access* 8 (2020), 61565–61575.

[25] Kam-Fai Chan and Dit-Yan Yeung. 2000. Mathematical expression recognition: a survey. *Int. J. on Doc. Anal. and Recog.* 3, 1 (2000), 3–15.

[26] Kam-Fai Chan and Dit-Yan Yeung. 2001. Error detection, error correction and performance evaluation in on-line mathematical expression recognition. *Pattern Recognition* 34, 8 (2001), 1671–1684.

[27] Kam-Fai Chan and Dit-Yan Yeung. 2001. PenCalc: A novel application of on-line mathematical expression recognition technology. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 774–778.

[28] Shi-Kuo Chang. 1970. A method for the structural analysis of two-dimensional mathematical expressions. *Information Sciences* 2, 3 (1970), 253–272.

[29] Qimin Cheng, Qian Zhang, Peng Fu, Conghuan Tu, and Sen Li. 2018. A survey and analysis on automatic image annotation. *Pattern Recognition* 79 (2018), 242–259.

[30] Anastasiia Cherneha, Dmytro Zhelezniakov, Pavlo Tytarchuk, and Vasyl Tereshchenko. 2021. Segmentation of Handwritten Mathematical Matrices Using the Area Voronoi Diagram. In *IEEE EUROCON 2021-19th International Conference on Smart Technologies*. IEEE, 107–112.

[31] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. 2018. State-of-the-art

speech recognition with sequence-to-sequence models. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.* 4774–4778.

[32] Eugene Cho. 2019. Hey Google, can I ask you something in private?. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–9.

[33] Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on information theory* 2, 3 (1956), 113–124.

[34] Philip A Chou. 1989. Recognition of equations using a two-dimensional stochastic context-free grammar. In *Visual Communications and Image Processing IV*, Vol. 1199. 852–865.

[35] Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. *arXiv preprint cmp-lg/9605012* (1996).

[36] Marco Cotogni, Claudio Cusano, and Antonino Nocera. 2021. Recursive Recognition of Offline Handwritten Mathematical Expressions. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 3138–3145.

[37] Hai Dai Nguyen, Anh Duc Le, and Masaki Nakagawa. 2015. Deep neural networks for recognizing online handwritten mathematical symbols. In *Proc. IEEE Asian Conf. on Pattern Recognition*. 121–125.

[38] Hai Dai Nguyen, Anh Duc Le, and Masaki Nakagawa. 2016. Recognition of online handwritten math symbols using deep neural networks. *IEICE Trans. on Information and Systems* 99, 12 (2016), 3110–3118.

[39] Illya Degtyarenko, Ivan Deriuga, Andrii Grygoriev, Serhii Polotskyi, Volodymyr Melnyk, Dmytro Zakharchuk, and Olga Radyvonenko. 2021. Hierarchical Recurrent Neural Network for Handwritten Strokes Classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2865–2869.

[40] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Int. Conf. Comp. Vision and Pattern Recog.* 248–255.

[41] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In *Proc. Int. Conf. on Machine Learning*. 980–989.

[42] Yannis A Dimitriadis and Juan Lopez Coronado. 1995. Towards an ART based mathematical editor, that uses on-line handwritten symbol recognition. *Pattern Recognition* 28, 6 (1995), 807–822.

[43] Haisong Ding, Kai Chen, and Qiang Huo. 2021. An Encoder-Decoder Approach to Handwritten Mathematical Expression Recognition with Multi-head Attention and Stacked Decoder. In *International Conference on Document Analysis and Recognition*. Springer, 602–616.

[44] Anh Duc Le. 2020. Recognizing Handwritten Mathematical Expressions via Paired Dual Loss Attention Network and Printed Mathematical Expressions. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*. 566–567.

[45] Michael Erdmann, Alexander Maedche, Hans-Peter Schnurr, and Steffen Staab. 2000. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In *Proceedings of the COLING-2000 Workshop on Semantic Annotation and Intelligent Content*. 79–85.

[46] Yuko Eto and Masakazu Suzuki. 2001. Mathematical formula recognition using virtual link network. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 762–767.

[47] Dingbang Fang and Chenhao Zhang. 2020. Multi-Feature Learning by Joint Training for Handwritten Formula Symbol Recognition. *IEEE Access* 8 (2020), 48101–48109.

[48] Ri-Chen Feng, Daw-Tung Lin, Ken-Min Chen, Yi-Yao Lin, and Chin-De Liu. 2019. Improving Deep Learning by Incorporating Semi-automatic Moving Object Annotation and Filtering for Vision-based Vehicle Detection. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. 2484–2489. https://doi.org/10.1109/SMC.2019.8914169

[49] David B Fogel. 2006. *Evolutionary computation: toward a new philosophy of machine intelligence*. Vol. 1. John Wiley & Sons.

[50] John Freeman. 1975. The modelling of spatial relations. *Computer graphics and image processing* 4, 2 (1975), 156–171.

[51] Benoît Frénay and Michel Verleysen. 2013. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems* 25, 5 (2013), 845–869.

[52] Utpal Garain and BB Chaudhuri. 2005. Segmentation of touching symbols for OCR of printed mathematical expressions: an approach based on multifactorial analysis. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 177–181.

[53] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.* 776–780.

[54] Alexandru-Lucian Georgescu, Horia Cucu, and Corneliu Burileanu. 2019. Progress on automatic annotation of speech corpora using complementary ASR systems. In *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 571–574.

[55] Felix A Gers and E Schmidhuber. 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks* 12, 6 (2001), 1333–1340.

[56] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.

[57] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with LSTM recurrent networks. *Journal of machine learning research* 3, Aug (2002), 115–143.

[58] David E Golberg. 1989. Genetic algorithms in search, optimization, and machine learning. *Addion wesley* 1989, 102 (1989), 36.

[59] E Mark Gold. 1967. Language identification in the limit. *Information and control* 10, 5 (1967), 447–474.

[60] Alex Graves. 2012. Connectionist temporal classification. In *Supervised Sequence Labelling with Recurrent Neural Networks*. 61–93.

[61] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proc. ACM Int. Conf. on Machine learning*. 369–376.

[62] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 6645–6649.

[63] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks* 18, 5-6 (2005), 602–610.

[64] Andrii Grygoriev, Illya Degtyarenko, Ivan Deriuga, Serhii Polotskyi, Volodymyr Melnyk, Dmytro Zakharchuk, and Olga Radyvonenko. 2021. HCRNN: a novel architecture for fast online handwritten stroke classification. In *International Conference on Document Analysis and Recognition*. Springer, 193–208.

[65] Wacef Guerfali and Réjean Plamondon. 1993. Normalizing and restoring on-line handwriting. *Pattern Recognition* 26, 3 (1993), 419–431.

[66] Benjamin Guthier, Kalun Ho, and Abdulmotaleb El Saddik. 2017. Language-independent data set annotation for machine learning-based sentiment analysis. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2105–2110. https://doi.org/10.1109/SMC.2017.8122930

[67] Jaekyu Ha, Robert M Haralick, and Ihsin T Phillips. 1995. Understanding mathematical expressions from document images. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*, Vol. 2. 956–959.

[68] Wenhao He, Yuxuan Luo, Fei Yin, Han Hu, Junyu Han, Errui Ding, and Cheng-Lin Liu. 2016. Context-aware mathematical expression recognition: An end-to-end framework and a benchmark. In *Proc. IEEE Int. Conf. on Pattern Recog.* 3246–3251.

[69] Pierre Héroux, Eugen Barbu, Sébastien Adam, and E Trupin. 2007. Automatic ground-truth generation for document image analysis and understanding. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*, Vol. 1. 476–480.

[70] Nina ST Hirata and Frank D Julca-Aguilar. 2015. Matching based ground-truth annotation for online handwritten mathematical expressions. *Pattern Recognition* 48, 3 (2015), 837–848.

[71] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[72] Zelin Hong, Ning You, Jun Tan, and Ning Bi. 2019. Residual birnn based seq2seq model with transition probability matrix for online handwritten mathematical expression recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 635–640.

[73] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 2001. Introduction to automata theory, languages, and computation. *Acm Sigact News* 32, 1 (2001), 60–65.

[74] Lei Hu and Richard Zanibbi. 2011. HMM-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 457–462.

[75] Lei Hu and Richard Zanibbi. 2013. Segmenting handwritten math symbols using adaboost and multi-scale shape context features. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1180–1184.

[76] Lei Hu and Richard Zanibbi. 2016. Line-of-sight stroke graphs and parzen shape context features for handwritten math formula representation and symbol segmentation. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 180–186.

[77] Lei Hu and Richard Zanibbi. 2016. MST-based visual parsing of online handwritten mathematical expressions. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 337–342.

[78] Bing Quan Huang, YB Zhang, and Mohand Tahar Kechadi. 2007. Preprocessing techniques for online handwriting recognition. In *Proc. IEEE Int. Conf. on Intel. Systems Design and Appl.* 793–800.

[79] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*. 4700–4708.

[80] Jesse F Hull. 1996. *Recognition of mathematics using a two-dimensional trainable context-free grammar*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[81] Frank Julca-Aguilar. 2016. *Recognition of online handwritten mathematical expressions using contextual information*. Ph.D. Dissertation.

[82] Frank JulcaAguilar, Nina ST Hirata, Christian ViardGaudin, Harold Mouchère, and Sofiane Medjkoune. 2014. Mathematical symbol hypothesis recognition with rejection option. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 500–505.

[83] Birendra Keshari and S Watt. 2007. Hybrid mathematical symbol recognition using support vector machines. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*, Vol. 2. 859–863.

[84] Viacheslav Khomenko, Andriy Volkoviy, Illya Degtyarenko, and Olga Radyvonenko. 2017. Handwriting Text/Non-Text Classification on Mobile Device. In *Proc. Int. Conf. on Art. Intel. and Pattern Recog.* 42.

[85] M Koschinski, H-J Winkler, and Manfred Lang. 1995. Segmentation and recognition of symbols within handwritten mathematical expressions. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*, Vol. 4. 2439–2442.

[86] Andreas Kosmala and Gerhard Rigoll. 1998. On-line handwritten formula recognition using statistical methods. In *Proc. IEEE Int. Conf. on Pattern Recog.*, Vol. 2. 1306–1308.

[87] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[88] Anand Kumar, A Balasubramanian, Anoop Namboodiri, and CV Jawahar. 2006. Model-based annotation of online handwritten datasets. In *Proc. IEEE Int. Workshop on Frontiers in Handwr. Recog.*

[89] P Pavan Kumar, Arun Agarwal, and Chakravarthy Bhagvati. 2014. A string matching based algorithm for performance evaluation of mathematical expression recognition. *Sadhana* 39, 1 (2014), 63–79.

[90] George Labahn, Edward Lank, Scott MacLean, Mirette Marzouk, and David Tausky. 2008. Mathbrush: A system for doing math on pen-based devices. In *Proc. Int. Workshop on Document Analysis Systems*. 599–606.

[91] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. Int. Conf. on Machine Learning*. 282–289.

[92] Adrien Lapointe and Dorothea Blostein. 2009. Issues in performance evaluation: A Case Study of Math Recognition. (2009), 1355–1359.

[93] Joseph J LaViola Jr and Robert C Zeleznik. 2004. MathPad 2: a system for the creation and exploration of mathematical sketches. *ACM Transactions on Graphics* 23, 3 (2004), 432–440.

[94] Stéphane Lavirotte and Loïc Pottier. 1998. Mathematical formula recognition using graph grammar. In *Document Recognition V*, Vol. 3305. 44–52.

[95] Anh Duc Le, Hai Dai Nguyen, and Masaki Nakagawa. 2016. Modified XY cut for re-ordering strokes of online handwritten mathematical expressions. In *Proc. IEEE Int. Workshop on Document Analysis Systems*. 233–238.

[96] Anh Duc Le, Bipin Indurkhya, and Masaki Nakagawa. 2019. Pattern generation strategies for improving recognition of handwritten mathematical expressions. *Pattern Recognition Letters* 128 (2019), 255–262.

[97] Anh Duc Le and Masaki Nakagawa. 2016. A system for recognizing online handwritten mathematical expressions by using improved structural analysis. *Int. J. on Doc. Anal. and Recog.* 19, 4 (2016), 305–319.

[98] Anh Duc Le and Masaki Nakagawa. 2017. Speedup of parsing for recognition of online handwritten mathematical expressions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*, Vol. 1. 896–901.

[99] Anh Duc Le and Masaki Nakagawa. 2017. Training an end-to-end system for handwritten mathematical expression recognition by generated patterns. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*, Vol. 1. 1056–1061.

[100] Hsi-Jian Lee and Jiumn-Shine Wang. 1997. Design of a mathematical expression understanding system. *Pattern Recognition Letters* 18, 3 (1997), 289–298.

[101] Stefan Lehmberg, H-J Winkler, and Manfred Lang. 1996. A soft-decision approach for symbol segmentation within handwritten mathematical expressions. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*, Vol. 6. 3434–3437.

[102] Willem JM Levelt. 2008. *An introduction to the theory of formal languages and automata*. John Benjamins Publishing.

[103] Zhe Li, Lianwen Jin, Songxuan Lai, and Yecheng Zhu. 2020. Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention. (2020), 175–180.

[104] Percy Liang, Mukund Narasimhan, Michael Shilman, and Paul Viola. 2005. Efficient geometric algorithms for parsing in two dimensions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1172–1177.

[105] Marcus Liwicki and Horst Bunke. 2009. Feature selection for HMM and BLSTM based handwriting recognition of whiteboard notes. *Int. J. Pattern Recognit. Artif. Intell.* 23, 05 (2009), 907–923.

[106] Marcus Liwicki, Alex Graves, Santiago Fernàndez, Horst Bunke, and Jürgen Schmidhuber. 2007. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*

[107] Arnaud Lods, Eric Anquetil, and Sébastien Macé. 2019. Fuzzy visibility graph for structural analysis of online handwritten mathematical expressions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 641–646.

[108] Scott MacLean and George Labahn. 2013. A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets. *Int. J. Doc. Anal. and Recog.* 16, 2 (2013), 139–163.

[109] Scott MacLean, George Labahn, Edward Lank, Mirette Marzouk, and David Tausky. 2011. Grammar-based techniques for creating ground-truthed sketch corpora. *Int. J. Doc. Anal. and Recog.* 14, 1 (2011), 65–74.

[110] Mahshad Mahdavi. 2020. Query-Driven Global Graph Attention Model for Visual Parsing: Recognizing Handwritten and Typeset Math Formulas. (2020).

[111] Mahshad Mahdavi and Richard Zanibbi. 2020. Visual Parsing with Query-Driven Global Graph Attention (QD-GGA): Preliminary Results for Handwritten Math Formula Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 570–571.

[112] Mahshad Mahdavi, Richard Zanibbi, Harold Mouchère, Christian Viard-Gaudin, and Utpal Garain. 2019. ICDAR 2019 CROHME+TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*

[113] Kim Marriott, Bernd Meyer, and Kent B Wittenburg. 1998. A survey of visual language specification and recognition. In *Visual language theory*. 5–85.

[114] MathBrush Labs. 2020. MathBrush. https://apps.apple.com/ca/app/mathbrush/id578957934. (Version 3.1.8) [Mobile app].

[115] Nicholas Elias Matsakis. 1999. *Recognition of handwritten mathematical expressions*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[116] Johannes Michael, Roger Labahn, Tobias Grüning, and Jochen Zöllner. 2019. Evaluating sequence-to-sequence models for handwritten text recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 1286–1293.

[117] Microsoft Corporation. 2020. Microsoft Math Solver. https://math.microsoft.com/. (Version 1.0.178) [Mobile app].

[118] Harold Mouchere, Christian Viard-Gaudin, D.H. Kim, J.H. Kim, and Utpal Garain. 2012. ICFHR 2012 competition on recognition of on-line mathematical expressions (CROHME 2012). In *Proc. IEEE Int. Workshop on Frontiers in Handwr. Recog.* 811–816.

[119] Harold Mouchere, Christian Viard-Gaudin, Dae Hwan Kim, Jin Hyung Kim, and Utpal Garain. 2011. CROHME 2011: Competition on recognition of online handwritten mathematical expressions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1497–1500.

[120] Harold Mouchere, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. 2014. ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014). In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 791–796.

[121] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. 2016. ICFHR 2016 CROHME: Competition on recognition of online handwritten mathematical expressions. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 607–612.

[122] Harold Mouchere, Christian Viard-Gaudin, Richard Zanibbi, Utpal Garain, Dae Hwan Kim, and Jin Hyung Kim. 2013. ICDAR 2013 CROHME: Third international competition on recognition of online handwritten mathematical expressions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1428–1432.

[123] Francisco Álvaro Muñoz. 2015. *Mathematical Expression Recognition based on Probabilistic Grammars*. Ph.D. Dissertation. Universitat Politécnica de Valéncia.

[124] Venkatesh N Murthy, Ethem F Can, and R Manmatha. 2014. A hybrid model for automatic image annotation. In *Proceedings of International Conference on Multimedia Retrieval*. 369–376.

[125] MyScript. 2020. MyScript Calculator 2. https://www.myscript.com/calculator. (Version 1.2.1) [Mobile app].

[126] Katsuhiko Nakamura and Masashi Matsumoto. 2002. Incremental learning of context free grammars. In *International Colloquium on Grammatical Inference*. Springer, 174–184.

[127] HD Nguyen, AD Le, and M Nakagawa. 2014. Combination of LSTM and CNN on recognizing mathematical symbols. *Information-Based Induction Sciences and Machine Learning* (2014), 287–292.

[128] Hung Tuan Nguyen, Cuong Tuan Nguyen, and Masaki Nakagawa. 2018. ICFHR 2018– Competition on Vietnamese Online Handwritten Text Recognition using HANDS-VNOnDB (VOHTR2018). In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 494–499.

[129] Akihiro Nomura, Kazuyuki Michishita, Seiichi Uchida, and Masakazu Suzuki. 2003. Detection and segmentation of touching characters in mathematical expressions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 126–130.

[130] Theresa O'Connell, Chuanjun Li, Timothy S Miller, Robert C Zeleznik, and Joseph J LaViola Jr. 2009. A usability evaluation of AlgoSketch: a pen-based application for mathematics. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*. 149–157.

[131] Emily Öhman. 2020. Challenges in Annotation: Annotator Experiences from a Crowdsourced Emotion Annotation Task.. In *DHN*. 293–301.

[132] Masayuki Okamoto. 1991. Recognition of mathematical expressions by using the layout structure of symbols. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 242–250.

[133] Masayuki Okamoto and Akira Miyazawa. 1992. An experimental implementation of a document recognition system for papers containing mathematical expressions. In *Structured Document Image Analysis*. 36–53.

[134] Oleg Okun and Matti Pietikainen. 2000. Automatic ground-truth generation for skew-tolerance evaluation of document layout analysis methods. In *Proc. Int. Conf. on Pattern Recognition*, Vol. 4. 376–379.

[135] Ling Ouyang and Richard Zanibbi. 2009. Identifying layout classes for mathematical symbols using layout context. In *Proc. IEEE Western New York Image Processing Workshop*.

[136] Khanh Minh Phan, Anh Duc Le, Bipin Indurkhya, and Masaki Nakagawa. 2018. Augmented incremental recognition of online handwritten mathematical expressions. *Int. J. Doc. Anal. and Recog.* 21, 4 (2018), 253–268.

[137] Solen Quiniou, Harold Mouchere, Christian Viard-Gaudin, Emmanuel Morin, Simon Petitrenaud, Sofiane Medjkoune, et al. 2011. HAMEX — a handwritten and audio dataset of mathematical expressions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 452–456.

[138] Janet C. Read, Stuart MacFarlane, and Chris Casey. 2002. Oops! Silly Me! Errors in a Handwriting Recognition-based Text Entry Interface for Children. In *Proceedings of the Second Nordic Conference on Human-computer Interaction* (Aarhus, Denmark) *(NordiCHI '02)*. ACM, New York, NY, USA, 35–40. https://doi.org/10.1145/572020.572026

[139] Stefano Crespi Reghizzi and Matteo Pradella. 2008. A CKY parser for picture grammars. *Inform. Process. Lett.* 105, 6 (2008), 213–217.

[140] Taik Heon Rhee and Jin Hyung Kim. 2009. Efficient search strategy in structural analysis for handwritten mathematical expression recognition. *Pattern Recognition* 42, 12 (2009), 3192–3201.

[141] Kunal Sain, Abhishek Dasgupta, and Utpal Garain. 2011. EMERS: a tree matching–based performance evaluation of mathematical expression recognition systems. *Int. J. Doc. Anal. and Recog.* 14, 1 (2011), 75–85.

[142] Yasubumi Sakakibara. 1992. Efficient learning of context-free grammars from positive structural examples. *Information and Computation* 97, 1 (1992), 23–60.

[143] Julian Salazar, Katrin Kirchhoff, and Zhiheng Huang. 2019. Self-attention networks for connectionist temporal classification in speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 7115–7119.

[144] Samsung Electronics Co., Ltd. 2020. S Note. https://galaxy.store/snot. (Version 5.2.07.14) [Mobile app].

[145] Jürgen Schmidhuber, F Gers, and Douglas Eck. 2002. Learning nonregular languages: A comparison of simple recurrent networks and LSTM. *Neural computation* 14, 9 (2002), 2039–2041.

[146] Jürgen Schmidhuber, Daan Wierstra, and Faustino J Gomez. 2005. Evolino: Hybrid neuroevolution/optimal linear search for sequence prediction. In *Proceedings of the 19th International Joint Conferenceon Artificial Intelligence (IJCAI)*.

[147] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[148] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.

[149] Clifford A Shaffer. 1997. *A practical introduction to data structures and algorithm analysis*. Prentice Hall Upper Saddle River, NJ.

[150] Fotini Simistira, Vassilis Katsouros, and George Carayannis. 2008. A template matching distance for recognition of on-line mathematical symbols. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.*

[151] Fotini Simistira, Vassilis Papavassiliou, Vassilis Katsouros, and George Carayannis. 2014. Recognition of spatial relations in mathematical formulas. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 164–168.

[152] Steven S Skiena. 2020. *The algorithm design manual*. Springer International Publishing.

[153] Steve Smithies, Kevin Novins, and James Arvo. 1999. A handwriting-based equation editor. In *Graphics Interface*, Vol. 99. 84–91.

[154] Jan Stria, Martin Bresler, Daniel Prua, and Václav Hlavác. 2012. MfrDB: Database of annotated on-line mathematical formulae. In *Proc. IEEE Int. Conf. on Frontiers in Handwr. Recog.* 542–547.

[155] Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori. 2003. INFTY: an integrated OCR system for mathematical documents. In *Proc. ACM symposium on Document engineering*. 95–104.

[156] Ernesto Tapia and Raúl Rojas. 2003. Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In *Proc. Int. workshop on graphics recognition*. 329–340.

[157] Ernesto Tapia and Raul Rojas. 2005. Recognition of on-line handwritten mathematical expressions in the e-chalk system-an extension. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1206–1210.

[158] Eugene M Taranta and Joseph J LaViola Jr. 2015. Math boxes: A pen-based user interface for writing difficult mathematical expressions. In *Proc. ACM Int. Conf. on Intelligent User Interfaces*. 87–96.

[159] Esma F Bilgin Tasdemir and Berrin Yanikoglu. 2019. A comparative study of delayed stroke handling approaches in online handwriting. *International Journal on Document Analysis and Recognition (IJDAR)* 22, 1 (2019), 15–28.

[160] Arit Thammano and Sukhumal Rugkunchon. 2006. A neural network model for online handwritten mathematical symbol recognition. In *Proc. Int. Conf. on Intell. Computing*. 292–298.

[161] William Thimbleby. 2004. A novel pen-based calculator and its evaluation. In *Proceedings of the third Nordic conference on Human-computer interaction*. 445–448.

[162] Xuedong Tian and Yan Zhang. 2007. Segmentation of touching characters in mathematical expressions using contour feature technique. In *Proc. IEEE Int. Conf. Software Eng., Artif. Intell., Network., Parallel/Distrib. Comput.*, Vol. 1. 206–209.

[163] Masaru Tomita. 1991. Parsing 2-dimensional language. In *Current issues in parsing technology*. 277–289.

[164] Seiichi Toyota, Seiichi Uchida, and Masakazu Suzuki. 2006. Structural analysis of mathematical formulae with verification based on formula description grammar. In *Proc. Int. Workshop on Document Analysis Systems*. 153–163.

[165] Kenichi Toyozumi, Naoya Yamada, Takayuki Kitasaka, Kensaku Mori, Yasuhito Suenaga, Kenji Mase, and Tomoichi Takahashi. 2004. A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information. In *Proc. IEEE Int. Conf. on Pattern Recog.*, Vol. 2. 630–633.

[166] Thanh-Nghia Truong, Hung Tuan Nguyen, Cuong Tuan Nguyen, and Masaki Nakagawa. 2021. Learning symbol relation tree for online mathematical expression recognition. *arXiv preprint arXiv:2105.06084* (2021).

[167] Andrew J Viterbi. 2006. A personal history of the Viterbi algorithm. *IEEE Signal Processing Magazine* 23, 4 (2006), 120–142.

[168] Valentyna Volkova, Ivan Deriuga, Vadym Osadchyi, and Olga Radyvonenko. 2018. Improvement of character segmentation using recurrent neural networks and dynamic programming. In *Proc. IEEE Int. Conf. on Data Stream Mining & Processing*. 218–222.

[169] Da-Han Wang, Fei Yin, Jin-Wen Wu, Yu-Pei Yan, Zhi-Cai Huang, Gui-Yun Chen, Yao Wang, and Cheng-Lin Liu. 2020. ICFHR 2020 Competition on Offline Recognition and Spotting of Handwritten Mathematical Expressions-OffRaSHME. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 211–215.

[170] Hongyu Wang and Guangcun Shan. 2020. Recognizing Handwritten Mathematical Expressions as LaTex Sequences Using a Multiscale Robust Neural Network. (2020). https://arxiv.org/abs/2003.00817 [Online].

[171] Jiaming Wang, Jun Du, Jianshu Zhang, Bin Wang, and Bo Ren. 2021. Stroke constrained attention network for online handwritten mathematical expression recognition. *Pattern Recognition* (2021), 108047.

[172] Jiaming Wang, Jun Du, Jianshu Zhang, and Zi-Rui Wang. 2019. Multi-modal Attention Network for Handwritten Mathematical Expression Recognition. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 1181–1186.

[173] Stephen M Watt, Tom Underhill, YM Chee, K Franke, M Froumentin, S Madhvanath, JA Magaña, G Pakosz, G Russell, M Selvaraj, et al. 2011. Ink markup language (InkML). *W3C Proposed Recommendation* 10 (2011).

[174] Wikipedia. 2021. Wikipedia:Database download — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Wikipedia%3ADatabase%20download&oldid=998637468. [Online; accessed 14-February-2021].

[175] Frauke Wilm, Christof A Bertram, Christian Marzahl, Alexander Bartel, Taryn A Donovan, Charles-Antoine Assenmacher, Kathrin Becker, Mark Bennett, Sarah Corner, Brieuc Cossic, et al. 2020. How Many Annotators Do We Need?–A Study on the Influence of Inter-Observer Variability on the Reliability of Automatic Mitotic Figure Assessment. *arXiv preprint arXiv:2012.02495* (2020).

[176] H-J Winkler. 1996. HMM-based handwritten symbol recognition using on-line and off-line features. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*, Vol. 6. 3438–3441.

[177] H-J Winkler, H Fahrner, and Manfred Lang. 1995. A soft-decision approach for structural analysis of handwritten mathematical expressions. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Vol. 4. 2459–2462.

[178] H-J Winkler and Manfred Lang. 1997. Online symbol segmentation and recognition in handwritten mathematical expressions. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*, Vol. 4. 3377–3380.

[179] Changjie Wu, Qing Wang, Jianshu Zhang, Jun Du, Jiaming Wang, Jiajia Wu, and Jinshui Hu. 2021. Stroke Based Posterior Attention for Online Handwritten Mathematical Expression Recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2943–2949.

[180] Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. 2018. Image-to-Markup Generation via Paired Adversarial Learning. In *Proc. Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*. 18–34.

[181] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. ACM Int. Conf. on Machine learning*. 2048–2057.

[182] Oleg Yakovchuk, Anastasiia Cherneha, Dmytro Zhelezniakov, and Viktor Zaytsev. 2020. Methods for Lines and Matrices Segmentation in RNN-based Online Handwriting Mathematical Expression Recognition Systems. In *2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP)*. IEEE, Lviv, Ukraine, 255–261.

[183] Ryo Yamamoto, Shinji Sako, Takuya Nishimoto, and Shigeki Sagayama. 2006. On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In *Proc. IEEE Int. Workshop on Frontiers in Handwr. Recog.*

[184] Koji Yatani and Khai N Truong. 2009. An evaluation of stylus-based text entry methods on handheld devices studied in different user mobility states. *Pervasive and Mobile Computing* 5, 5 (2009), 496–508.

[185] Richard Zanibbi and Dorothea Blostein. 2012. Recognition and retrieval of mathematical expressions. *Int. J. Doc. Anal. and Recog.* 15, 4 (2012), 331–357.

[186] Richard Zanibbi, Dorothea Blostein, and James R. Cordy. 2002. Recognizing mathematical expressions using tree transformation. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 11 (2002), 1455–1467.

[187] Richard Zanibbi, Amit Pillay, Harold Mouchere, Christian Viard-Gaudin, and Dorothea Blostein. 2011. Stroke-based performance metrics for handwritten mathematical expressions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 334–338.

[188] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

[189] Robert Zeleznik, Timothy Miller, Chuanjun Li, and Joseph J LaViola. 2008. MathPaper: Mathematical sketching with fluid support for interactive computation. In *International Symposium on Smart Graphics*. 20–32.

[190] J. Zhang, J. Du, and L. Dai. 2017. A GRU-based encoder-decoder approach with attention for online handwritten mathematical expression recognition. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*, Vol. 1. 902–907.

[191] Jianshu Zhang, Jun Du, and Lirong Dai. 2018. Multi-scale attention with dense encoder for handwritten mathematical expression recognition. In *Proc. IEEE Int. Conf. on Pattern Recog.* 2245–2250.

[192] J. Zhang, J. Du, and L. Dai. 2018. Track, Attend, and Parse (TAP): An End-to-End Framework for Online Handwritten Mathematical Expression Recognition. *IEEE Trans. on Multimedia* 21, 1 (2018), 221–233.

[193] Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, and Lirong Dai. 2017. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition* 71 (2017), 196–206.

[194] Ling Zhang, Dorothea Blostein, and Richard Zanibbi. 2005. Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.* 972–976.

[195] Ting Zhang. 2017. *New Architectures for Handwritten Mathematical Expressions Recognition*. Ph.D. Dissertation. Université de Nantes.

[196] Ting Zhang, Harold Mouchère, and Christian Viard-Gaudin. 2016. Using BLSTM for interpretation of 2-D languages. *Document numérique* 19, 2 (2016), 135–157.

[197] Ting Zhang, Harold Mouchere, and Christian Viard-Gaudin. 2017. Tree-based BLSTM for mathematical expression recognition. In *Proc. IEEE Int. Conf. on Doc. Anal. and Recog.*, Vol. 1. 914–919.

[198] Wenqi Zhao, Liangcai Gao, Zuoyu Yan, Shuai Peng, Lin Du, and Ziyin Zhang. 2021. Handwritten Mathematical Expression Recognition with Bidirectionally Trained Transformer. *arXiv preprint arXiv:2105.02412* (2021).

[199] Dmytro Zhelezniakov, Anastasiia Cherneha, Viktor Zaytsev, Tetiana Ignatova, Olga Radyvonenko, and Oleg Yakovchuk. 2020. Evaluating new requirements to pen-centric intelligent user interface based on end-to-end mathematical expressions recognition. In *Proc. ACM Int. Conf. on Intelligent User Interfaces*. 212–220.

[200] Dmytro Zhelezniakov, Viktor Zaytsev, and Olga Radyvonenko. 2021. Online Handwritten Mathematical Expression Recognition and Applications: A Survey. *IEEE Access* 9 (2021), 38352–38373.

[201] Dmytro Zhelezniakov, Viktor Zaytsev, Olga Radyvonenko, and Yevhenii Yakishyn. 2019. InteractivePaper: Minimalism in Document Editing UI Through the Handwriting Prism. In *The Adjunct Publication of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 13–15.

[202] Matthias Zimmermann and Horst Bunke. 2002. Automatic segmentation of the IAM off-line database for handwritten English text. In *Object recognition supported by user interaction for service robots*, Vol. 4. IEEE, 35–39.

[203] Анатолій Васильович Анісімов, Василь Миколайович Терещенко, and І.В. Кравченко. 2002. Основні алгоритми обчислювальної геометрії: Навч. посібн. *К.: Київський університет* (2002).

[204] Сергій Олександрович Субботін. 2020. Нейронні мережі: теорія та практика.

Додаток А

Участь у міжнародних змаганнях

A.1. ICDAR 2019: Competition on Recognition of Handwritten Mathematical Expressions - CROHME

У 2019 році запропонована система брала участь у конкурсі з розпізнавання рукописних математичних виразів – CROHME [112]. Організатори опублікували набір даних для навчання та верифікації, який складався з близько 8 500 прикладів. Однак учасники не були обмежені лише цими наборами і могли використовувати додаткові дані для тренування моделей. Деякі учасники синтетично збільшували розмір навчального набору даних. Розмір алфавіту MB становив 101 символ.

Необхідно відзначити, що в цьому змаганні бралися до уваги тільки якісні характеристики систем, а швидкісні характеристики не розглядалися. Тому більшість учасників представили рішення, які вимагають потужні обчислювальні ресурси. Для прикладу, рішення, яке зайняло перше місце, побудовано за допомогою об'єднання декількох моделей в режимі онлайн і офлайн та нейронної моделі мови. При цьому кожна модель окремо мала кілька мільйонів ваг.

У цьому конкурсі була представлена система, яка ґрунтується на підходах, запропонованих у дисертаційній роботі. На той час деякі елементи системи ще не були реалізовані. Зокрема, був відсутній додатковий класифікатор для односимвольних виразів. Навчання моделей здійснювалося на наборі даних з розміром алфавіту в 101 символ, який був сформований з відкритого набору CROHME і закритого набору даних. Спеціально для участі в конкурсі була підготовлена спрощена граматика, яка відповідала набору виразів з навчального набору даних. Також використовувався випадковий ліс замість дерева рішень для класифікації просторових відношень.

Запропонований варіант системи посів друге місце з точністю розпізнавання виразів на рівні 79.82% та точністю розпізнавання дерева виразу на рівні 89.32%. Результати змагання представлені в Таблиці А.2.

Таблиця А.1. ICDAR 2019: офіційні результати змагання

Team	Structure + Symbol Labels			Structure
	Correct	$\leqslant$ 1 s.err	$\leqslant$ 2 s.err	Correct
USTC-iFLYTEK	80.73	88.99	90.74	91.49
Samsung R&D 1	**79.82**	**87.82**	**89.15**	**89.32**
MyScript	79.15	86.82	89.82	90.66
Sun Yat-Sen U.	77.40	85.82	87.99	88.82
Samsung R&D 2	65.97	77.81	81.73	82.82
PAL-v2	62.55	74.98	78.40	79.15
MathType	60.13	74.40	78.57	79.15
TUAT	39.95	52.21	56.54	58.22

Таблиця А.2. ICDAR 2019: офіційні результати змагання (задача 1B: штрихи з відомими символами)

Team	Structure + Symbol Labels			Structure
	Correct	$\leqslant$ 1 s.err	$\leqslant$ 2 s.err	Correct
Samsung R&D 1	**92.94**	**93.11**	**93.20**	**93.20**
fvg	85.88	85.88	85.96	85.88

A.2. ICFHR 2020: Competition on Offline Recognition and Spotting of Handwritten Mathematical Expressions - OffRaSHME

Наступне змагання з офлайн розпізнавання та виявлення РМВ [169] відбувалося у 2020 році. Так само, як і в змаганні з онлайн розпізнання, тут також не бралися до уваги швидкісні характеристики. Тому більшість представлених рішень побудовані на нейронних мережах з величезною кількістю ваг та за допомогою об'єднання декількох моделей.

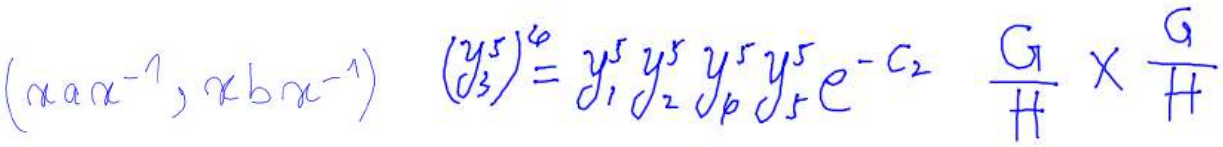
Для участі був додатково розроблений модуль із трансформації зображення в набір штрихів на базі ідей, представлених у роботі [24]. Моделі були повністю підготовлені із зразків, отриманих за допомогою алгоритму трансформації зображення в набір штрихів. Запропонована система показала якість розпізнавання на рівні 61.90% (Таблиця А.3). Результат був отриманий на єдиній моделі з сумарною кількістю параметрів близько 300 000. Точність розпізнавання в такій системі дуже залежить від алгоритму вилучення штрихів із зображення.

Таблиця А.3. ICFHR 2020: офіційні результати змагання (з додатковими наборами даних для тренування)

	Correct	$\leqslant$ 1 s.err	$\leqslant$ 2 s.err
USTC-iFLYTEK	81.85	90.50	92.30
IVTOV	**61.90**	**73.00**	**75.95**
MLE	46.85	61.15	65.80

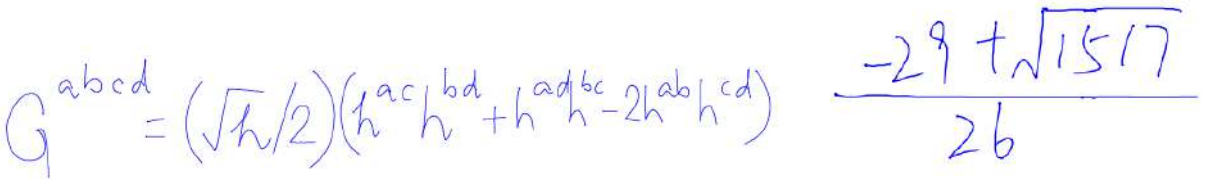
Додаток Б

Приклади роботи системи розпізнавання математичних виразів

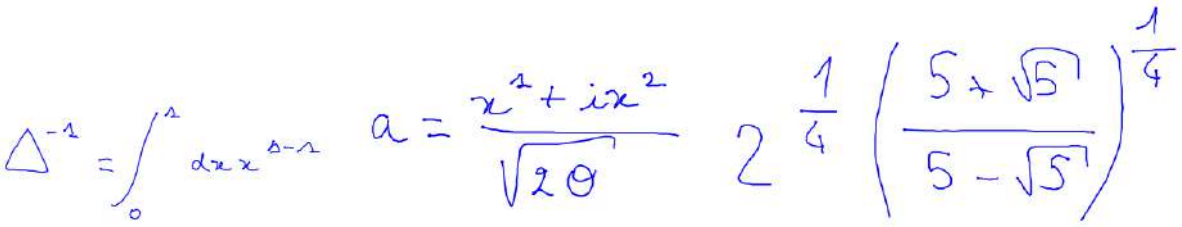


(а) (xax^{-1}, xbx^{-1}) (б) $(y_3^5)^4 = y_1^5 y_2^5 y_4^5 y_5^5 e^{-c_2}$ (в) $\frac{G}{H} \times \frac{G}{H}$

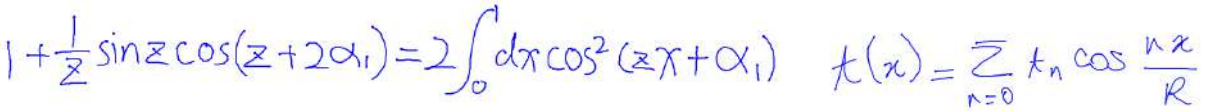
(г) $(\cos(z) - 1)/z^2, \sin(z)/z, (\sin(z) - z)/z^3$ (д) $c = \frac{3k}{k+2} - 1 = \frac{2(k-1)}{k+2}$



(е) $G^{abcd} = (\sqrt{h}/2)(h^{ac}h^{bd} + h^{ad}h^{bc} - 2h^{ab}h^{cd})$ (ж) $\frac{-29+\sqrt{1517}}{26}$

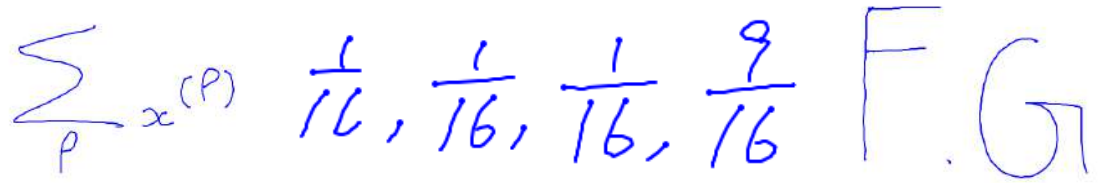


(и) $\Delta^{-1} = \int_0^1 dx x^{\Delta-1}$ (к) $a = \frac{x^1+ix^2}{\sqrt{2\theta}}$ (л) $2^{\frac{1}{4}} \left(\frac{5+\sqrt{5}}{5-\sqrt{5}}\right)^{\frac{1}{4}}$



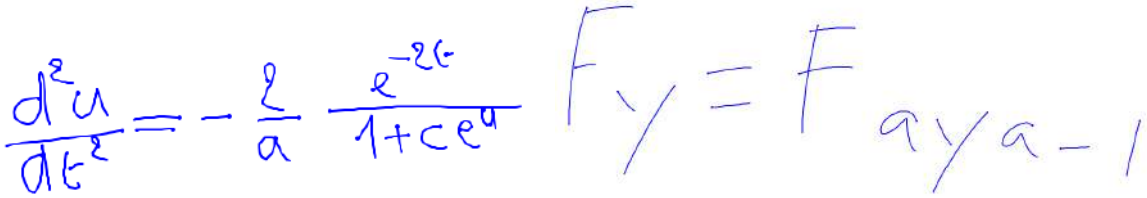
(м) $1 + \frac{1}{z}\sin z \cos(z + 2\alpha_1) = 2\int_0^1 dx \cos^2(zx + \alpha_1)$ (н) $t(x) = \sum_{n=0} t_n \cos\frac{nx}{R}$

Рис. Б.1. Приклади рукописних виразів, які були правильно розпізнані



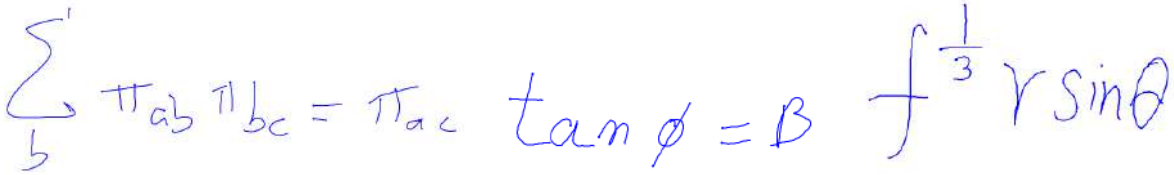
(а) $\sum_{l} x^{(l)} \to \sum_{p} x^{(p)}$ (б) $\frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{9}{16} \to \frac{1}{1l}, \frac{1}{16}, \frac{1}{16}, \frac{9}{16}$ (в) $F.G \to F \cdot G$

(г) $\sin(t) \to sm(t)$ (д) $-b \leqslant x \leqslant b \to -6 \leqslant x \leqslant 6$ (е) $\sigma\sigma\sigma \to 000$



(ж) $\frac{d^2u}{dt^2} = -\frac{2}{a}\frac{e^{-2t}}{1+ce^u} \to \frac{d^2u}{dt^2} = -\frac{2}{a}\frac{e^{-2c}}{1+ce^u}$ (и) $F_y = F_{aya^{-1}} \to F_y = F_{aya-1}$

(к) $X_z = X_1 + iX_2 \to x_2 = x_1 + ix_2$ (л) $\lim_{r\to\infty} e^{2r}(n - H(r)) = 2n \to \lim_{r\to\infty} e^{2^r}(n - H(r)) = 2n$



(м) $\sum_{b} \pi_{ab}\pi_{bc} = \pi_{ac} \to \sum_{b}^{1} \pi_{ab}1b_c = \pi_{ac}$ (н) $\tan\phi = B \to \tan\phi = \beta$ (п) $f\frac{1}{3}r\sin\theta \to f\frac{1}{3}r\sin\theta$

(р) $t_1(t) = -t_2(t) = t^{n+\frac{1}{2}} \to t_1(t) = -t_2(t) = t^n + \frac{1}{2}$ (с) $x^7x^8x^9 \to x^7x^8x^a$ (т) $2c \geq \sqrt{6} - \sqrt{3} \to 2c > \sqrt{6^1} - \sqrt{3^7}$

Рис. Б.2. Приклади рукописних виразів, які були розпізнані неправильно

Документ підписано у сервісі Вчасно (продовження)
thesis_zhelezniakov_signed_compressed.pdf

Документ відправлено: 14:30 03.11.2022

Власник документу

Електронний підпис
14:30 03.11.2022
Ідентифікаційний код: 2837601697
Железняков Дмитро Валентинович
Власник ключа: Железняков Дмитро Валентинович
Час перевірки КЕП/ЕЦП: 14:30 03.11.2022
Статус перевірки сертифікату: Сертифікат діє
Серійний номер: 3ED5083160DBC59B04000000A2B011001687AB00
Тип підпису: кваліфікований