УДК 004.89
DOI 10.18822/byusu20240351-56

ПОДГОТОВКА ДАННЫХ ДАТАСЕТА CROHME 2019 ДЛЯ ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ

Исмагулов Милан Ерикович
аспирант 2 года обучения,
Югорский государственный университет,
Ханты-Мансийск, Россия
E-mail: m_ismagulov@ugrasu.ru

Предмет исследования: в данной статье рассматривается процесс подготовки данных для обучения нейросетевых моделей, решающих задачу распознавания сложных рукописных математических выражений и их преобразования в системы компьютерной вёрстки, такие как LaTeX или MathML. В качестве исследуемого объекта используется датасет CROHME 2019, содержащий изображения рукописных математических выражений и их текстовые аннотации в формате LaTeX.

Цель исследования: разработка и описание процесса подготовки данных, который обеспечит корректное обучение модели и минимизирует ошибки в числовых метках и преобразованиях. В рамках исследования подробно рассматриваются следующие этапы: загрузка и предобработка изображений, парсинг LaTeX-аннотаций, связывание изображений с соответствующими аннотациями, нормализация значений пикселей, создание и проверка словаря токенайзера, а также подготовка самого токенайзера.

Методы включают анализ и обработку данных с использованием специализированных инструментов для работы с изображениями и текстовыми аннотациями.

Основным результатом работы является формирование датасета, готового к использованию в процессе обучения нейросетевой модели. Подготовленный датасет обеспечивает точное соответствие между изображениями и аннотациями, а также корректное преобразование математических выражений в LaTeX-код.

Ключевые слова: подготовка данных, датасет CROHME 2019, рукописные математические выражения, распознавание символов, компьютерная верстка, LaTeX, токенайзер, машинное обучение, препроцессинг изображений, парсинг аннотаций, масштабирование данных, нейронные сети, обработка изображений, словарь токенайзера.

DATASET PREPARATION FOR CROHME 2019 FOR TRAINING A NEURAL NETWORK

Milan E. Ismagulov
second-year PhD student,
Yugra State University,
Khanty-Mansiysk, Russia
E-mail: m_ismagulov@ugrasu.ru

Subject of research. This article examines the process of data preparation for training neural network models that address the task of recognizing complex handwritten mathematical expressions and converting them into typesetting systems like LaTeX or MathML. The dataset under study is CROHME 2019, which contains images of handwritten mathematical expressions and their corresponding LaTeX annotations.

The objective of this research is to develop and describe a data preparation process that ensures correct model training and minimizes errors in numerical labels and transformations. The study focuses on the following key stages: loading and preprocessing images, parsing LaTeX annotations, linking images with their corresponding annotations, normalizing pixel values, creating and verifying the tokenizer vocabulary, and preparing the tokenizer itself.

The methods include data analysis and processing using specialized tools for working with images and text annotations.

The main outcome of this work is the creation of a dataset ready for use in training the neural network model. The prepared dataset ensures accurate alignment between images and annotations, as well as correct conversion of mathematical expressions into LaTeX code.

Keywords: data preparation, CROHME 2019 dataset, handwritten mathematical expressions, symbol recognition, typesetting, LaTeX, tokenizer, machine learning, image preprocessing, annotation parsing, data scaling, neural networks, image processing, tokenizer dictionary.

ВВЕДЕНИЕ

Датасет CROHME – результат многолетних наработок для одноименного конкурса. Название представляет собой аббревиатуру Competition on Recognition of Handwritten Mathematical Expressions – «соревнование по распознаванию рукописных математических выражений». Датасет содержит 22 024 тренировочных изображения и 5 255 тестовых, столько же в датасете представлено аннотаций в формате InkML. InkML – это формат файла, основанный на XML, содержит информацию о LaTeX-аннотациях, трассеры движения пера для онлайн-распознавания математических выражений. Также в датасете присутствуют файлы формата SymLG в таком же количестве, представляющие собой файлы, описывающие каждый знак в формуле, его положение и отношение между знаками.

В качестве основы были взяты данные датасета CROHME за 2019 год, поскольку эти данные стандартизированы по размеру, а именно 1010 на 1010 пикселей. Данные по остальным годам необходимо дополнительно обрабатывать, обрезать по содержимому или приводить все к одному размеру. Данные за 2019 год представлены в количестве 10 979 тренировочных и 1 199 тестовых. Условно подготовку датасета CROHME можно разделить на следующие этапы:

1. Загрузка и препроцессинг изображений.

2. Парсинг аннотаций из файлов InkML и объединение загруженных данных в пары (изображение, аннотация).

3. Объединение данных в массивы NumPy и масштабирование диапазона пикселей из [0; 255] до [0; 1].

4. Создание словаря токенайзера, а также настройка самого токенайзера и проверка корректности работы токенайзера методом обратной токенизации.

5. Сравнение оригинальных аннотированных формул и обратно токенизированных для проверки полноты словаря и оценка полноты датасета по словарю.

Подробно каждый из пунктов рассмотрен в тексте статьи.

РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

Целью данной работы является разработка и реализация процесса подготовки данных для обучения нейросетевой модели, способной распознавать сложные рукописные математические выражения и преобразовывать их в формат LaTeX. Основные задачи включают предобработку изображений, корректное связывание изображений с аннотациями, нормализацию данных, создание и верификацию словаря токенайзера, а также подготовку токенайзера для дальнейшего использования в процессе обучения модели.

Загрузка и препроцессинг изображений

Препроцессинг (предобработка) изображений – это этап обработки данных, направленный на подготовку изображений к дальнейшему анализу и использованию в алгоритмах машинного обучения. Целью препроцессинга является улучшение качества изображений и приведение их к форме, которая наиболее подходит для последующего анализа и распознавания [1]. Может включать в себя следующие этапы:

1. Нормализация яркости и контрастности и бинаризация.
2. Удаление шумов и артефактов на изображениях.
3. Утолщение контуров изображения.
4. Изменение размера изображения.
5. Ротация и выравнивание.
6. Аугментация данных.

В случае обработки изображений датасета CROHME 2019 была задействована OpenCV [9] – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. В качестве средств предобработки были задействованы следующие функции:

1. Размытие по Гауссу – является распространенным методом обработки изображений, который используется для уменьшения шума и деталей в изображении. Оно осуществляется путем применения гауссовского фильтра, который является сверткой изображения, основанной на функции Гаусса [5, 6]:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},$$

где x, y – координаты пикселей изображения; σ – стандартное отклонение.

Процесс гауссового размытия заключается в том, что каждому пикселю изображения присваивается взвешенное среднее значение его собственных значений и значений соседних пикселей. Веса задаются гауссовой функцией, которая придает больший вес пикселям, расположенным ближе к рассматриваемому пикселю, и меньший вес тем, что дальше. Это приводит к сглаживанию изображения и уменьшению шума, сохраняя при этом основные структуры и формы [2]. В библиотеке OpenCV гауссово размытие вызывается с помощью следующей команды: cv2.GaussianBlur.

2. Также для обработки изображения была задействована функция адаптивной бинаризации [3] – cv2.adaptiveThreshold. Эта функция позволяет преобразовать изображение в черно-белое, используя локальные области для вычисления пороговых значений, что делает ее полезной для изображений с неравномерным освещением или тенями. В качестве адаптивного метода было использовано cv2.ADAPTIVE_THRESH_GAUSSIAN_C: Взвешенное среднее значение соседних пикселей с использованием гауссова окна. Тип бинаризации cv2.THRESH_BINARY: если значение пикселя больше порогового, присваивается maxValue, иначе 0.

3. Последним этапом будет изменение размера изображения с 1010 x 1010 пикселей до 256 x 256 пикселей, если изображения не уменьшить, то на этапе создания массива NumPy возникает ошибка «MemoryError», поскольку массив, состоящий из 10 тысяч изображений 1010 на 1010 пикселей, будет иметь высокие требования к объему оперативной памяти вычислительного устройства из-за чрезмерного объема данных.

Парсинг аннотаций LaTeX из файлов InkML

InkML – называемое пространство имен стандарта XML-файлов имеет расширение файла InkML, содержит в себе информацию об аннотациях двух форматов компьютерной верстки, а именно LaTeX и MathML, а также координаты трассировки пера по планшету или другому захватывающему устройству.

Пример содержимого InkML-файла

```
<ink xmlns="http://www.w3.org/2003/InkML">
<traceFormat>
<channel name="X" type="decimal"/>
<channel name="Y" type="decimal"/>
</traceFormat>
<annotation type="truth"> $y = Ax + A^2$</annotation>
<annotation type="UI">2012_IVC_CROHME_F01_E000</annotation>
<annotation type="copyright">LUNAM/IRCCyN</annotation>
<annotation type="writer">CROHME01</annotation>
<annotationXML type="truth" encoding="Content-MathML">
<math xmlns='http://www.w3.org/1998/Math/MathML'>
<mrow>
<mi xml:id="y_1">y</mi>
<mrow>
<mo xml:id="=_1">=</mo>
<mrow>
<mi xml:id="A_1">A</mi>
</mrow>
</mrow>
</mrow>
</math>
</annotationXML>
<trace id="0">
1.0066 5.46535, 1.0066 5.4645,...
</trace>
<traceGroup xml:id="11">
<annotation type="truth"> Segmentation</annotation>
<traceGroup xml:id="12">
<annotation type="truth">y</annotation>
<traceView traceDataRef="0"/>
<annotationXML href="y_1"/>
</traceGroup>
</traceGroup>
</ink>
```

Файл содержит в себе следующие составляющие:

1. Открывающий тег с ссылкой на xmlns (XML namespaces) <ink></ink>.

2. Тег <traceFormat></traceFormat> содержит в себе описание 2 каналов X и Y, type="decimal" говорит нам о том, что координаты записаны в десятичных числах.

3. Первый тег аннотации с типом, равным truth (истина), содержит в себе LaTeX-аннотацию.

4. Тег <annotation type="UI">2012 _IVC_ CROHME_F01_E000</annotation> содержит в себе информацию об уникальном идентификаторе отдельного InkML с указанием года создания, типа информации, конкурса и порядковых номеров в каталоге.

5. Аннотация с типом copyright описывает правообладателя, в данном случае это университет Нанта во Франции.

6. Следующая аннотация описывает автора данного файла.

7. Тег <annotationXML> </annotationXML> описывает истинное значение InkML-файла в формате компьютерной верстки MathML.

8. Тег <trace></trace> описывает координаты трассера электронного пера, например графического планшета, через пробел запи-

сывается координата X и Y, а через запятую – отдельные точки, с помощью id-номера создается ссылка внутри файла на знак, описываемый трассировкой в теге traceGroup, внутри которого находится другой тег traceView с атрибутом traceDataRef="0". С помощью этой конструкции мы объединяем описание конкретного знака компьютерной верстки с трассировкой, в нашей задаче трассировка напрямую не используется, поскольку на вход нейронной сети подается изображение, а не координаты точек, однако с помощью утилиты CROHMELib трассеры могут быть преобразованы в растровые изображения.

9. Тег <traceGroup></traceGroup> содержит в себе информацию об отдельных знаках, которые составляют целую формулу, внутри тега есть информация в формате LaTeX и MathML с атрибутом truth, а также ссылку на трассер.

В нашей задаче для создания пар [изображение; аннотация] из InkML-файла достаточно парсить LaTeX-аннотации с атрибутом truth; затем, чтобы объединить их, создадим функцию на скриптовом языке программирования Python, в скрипте в качестве ключа будет использовано имя файла изображения и InkML, поскольку имена совпадают, далее выведем первые 5 пар, пример пары показан на рисунке 1.

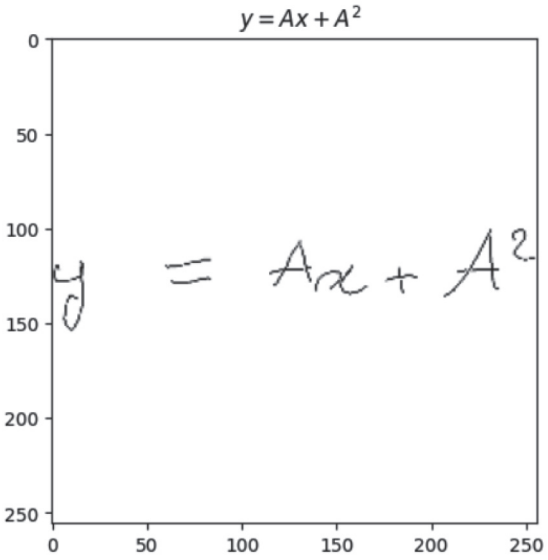

Рисунок 1. Пример готовой пары [изображение; аннотация]

На данном рисунке мы можем наблюдать сверху формулу, полученную с помощью LaTeX, и изображение, оси координат демонстрируют размер изображения в пикселях, в данном случае 256 на 256 пикселей, видна также работа функции гауссова размытия.

Объединение данных в массив NumPy и масштабирование диапазона пикселей

NumPy – это библиотека Python для работы с большими и многомерными массивами, большинство методов и функций библиотеки написано на Fortran и Си, что позволяет с большой ресурсной эффективностью и скоростью обрабатывать большие данные.

Объединение данных таким образом является классическим для машинного обучения. Напоминаем, что в датасете присутствует около 11 тысяч тренировочных и почти 1200 тестовых изображений, столько же InkML-аннотаций, что по умолчанию подразумевает использование NumPy-массивов. Иначе создать такой массив методами языка Python будет очень ресурсозатратно и медленно.

В свою очередь масштабирование диапазона пикселей – это процесс обработки данных изображения со стандартного диапазона [0; 255], где 0 – это черный цвет, а 255 – белый, до [0; 1] соответственно. Данный шаг способствует стабилизации работы алгоритмов машинного обучения и относится к методам оптимизации, подобный шаг положительно сказывается на скорости и стабильности процесса обучения, нормализация данных подобным образом положительно сказывается на проблеме «исчезающих» и «взрывающихся» градиентов, уменьшает разброс данных и гармонизирует их, что хорошо для алгоритма градиентного спуска.

Можно выделить 3 метода масштабирования диапазона float16, float32 и float64, где float говорит нам о том, что это числа с плавающей точкой, 16, 32, 64 – это разрядность в битах.

В нашей работе задействован метод float32. Этот выбор был сделан руководствуясь принципом «золотой середины».

Поскольку float16 хоть и очень быстро работает, в современных вычислениях он применяется редко из-за потерь большого количества информации, а float64 в разы увеличивает требования по ресурсам памяти компьютера.

Создание словаря токенайзера и настройка токенайзера

Процесс создания словаря токенайзера и его токенизация – это важнейшая часть настройки датасета для обучения модели, поскольку задача связана с преобразованием двумерного изображения с математической формулой в одномерную LaTeX-последовательность, именно для последней и необходимо создание токенайзера. Создание токенайзера условно можно поделить на 2 этапа: первый этап представляет собой создание массива со всеми элементами синтаксиса LaTeX-кода. В него входят:

1. Цифры от 0 до 9.
2. Латинский алфавит в нижнем регистре.
3. Латинский алфавит в верхнем регистре.
4. Греческий алфавит в нижнем регистре.
5. Уникальные буквы греческого алфавита в верхнем регистре (поскольку часть из букв присутствует в латинском алфавите, то, например, альфа и бета в верхнем регистре эквивалентны A и B в латинском).
6. Уникальные знаки синтаксиса LaTeX, например: знаки доллара, прямой слеш, обратный слеш, квадратные и фигурные скобки, операторы сравнения и логические операторы.
7. Операторы, представляющие собой целое слово синтаксиса LaTeX.

Второй этап представляет собой создание словаря, состоящего из двух компонент [ключ (в качестве ключа выступают элементы массива, созданного в предыдущем шаге); числовое значение]. Итоговый вид словаря будет таким:

Python-код (JSON-формат)

```
latex_tokens = {
        “\\frac”: 1,
        "\\parallel": 2,
"\\alpha": 3,
"\\pi": 4,
"\\beta": 5,
"=",
...
"\\leq": 283,
}
```

В качестве алгоритма токенизации был выбран алгоритм из библиотеки Tensorflow Keras и настроен таким образом, чтобы работать с готовым словарем, поскольку по умолчанию этот токенайзер работает только с текстом. Для настройки токенайзера на словарь LaTeX-выражения используется функция “re.findall” из библиотеки Regex (сокр. от Regular Expressions), которая работает с регулярными выражениями, описываемыми в словаре LaTeX-синтаксиса. Функция проверяет аннотации из файла InkML со значениями в словаре и разбивает формулу на отдельные составляющие и каждому из них присваивает номер. Например, формула:

$y = Ax + A^2$

токенизируется в:

['$', 'y', '=', 'A', 'x', '+', 'A', '^', '2', '$'].

Затем кодируется в числовые метки:

[40, 101, 5, 103, 100, 29, 103, 21, 69, 40].

Последние два этапа настройки токенайзера – это One-Hot Encoding и приведение последовательностей к одинаковой длине. Первое используется для выравнивания категорий, чтобы избежать случаев создания так называемых «ложных приоритетов», когда модель часто встречает объекты определенного класса. One-Hot Encoding выравнивает процесс обучения. Приведение последовательностей к одинаковой длине необходимо для повышения эффективности обучения, поскольку, например, RNN-нейросети лучше работают с последовательностями одинаковой (предсказуемой) длины [4]. В нашей работе это достигается путем выбора максимальной длины исходя из самой длинной формулы плюс два знака. Далее, если длина остальных формул не достигает максимальной длины, остальная часть вектора заполняется нулями.

К примеру, у нас есть три вектор-строки разной длины:

[1, 2, 4]
[2, 11, 15, 8, 72, 67]
[10, 24, 31, 17].

В данном случае максимальная размерность вектор-строки будет 6 + 2, если обратить внимание на вторую вектор-строку, приведя к одинаковой длине, получим:

[1, 2, 4, 0, 0, 0, 0, 0]
[2, 11, 15, 8, 72, 67, 0, 0]
[10, 24, 31, 17, 0, 0, 0, 0].

Теперь размерность каждой вектор-строки равна 8.

Для проверки корректности работы токенайзера необходимо произвести обратную токенизацию – это метод, при котором закодированные последовательности чисел в каждой вектор-строке с помощью словаря преобразуются обратно в формулы, полученные

от обратной токенизации. Данные можно использовать для оценки полноты словаря и датасета.

Проверка полноты словаря и датасета

Для проверки полноты словаря был создан скрипт, который построчно проверяет оригинальные аннотированные формулы и формулы, которые были получены методом обратной токенизации. Они должны совпадать. Если словарь неполный, то в обратно токенизированной формуле будет отсутствовать необходимый оператор, в таком случае оригинал формулы и обратно токенизированная формула записываются в текстовый файл для сравнения. Затем, проанализировав, можно дополнить словарь отсутствующим оператором.

Также необходимо произвести проверку тех меток, что будут поданы на вход нейронной сети. Если какая-либо метка есть в словаре, но отсутствует в аннотациях датасета, она игнорируется. Так, проверив полноту датасета CROHME 2019, было выяснено, что из 283 меток, которые есть в словаре, 154 из них отсутствуют в датасете, и только 129 присутствуют. Большая часть LaTeX-операторов в датасете не представлена.

ЗАКЛЮЧЕНИЕ И ВЫВОДЫ

По результатам проверки полноты датасета было выявлено, что из 283 значений словаря LaTeX в датасете представлено 129, 154 было утеряно. Это позволяет сделать вывод о том, что обучение нейронной сети по распознаванию рукописных математических выражений только на данных за 2019 год нецелесообразно, поскольку обучение на неполных данных может привести к ситуации, когда сеть будет некорректно распознавать отдельные математические выражения.

Рекомендуется дополнить данные за 2019 год датасетами CROHME за другие годы, доработать скрипт препроцессинга изображений, поскольку данные предыдущих годов имеют существенные различия в размерах изображений, например в данных за 2013 год все изображения имеют разный размер и разрешение.

Также следует отойти от концепции заранее разделенных данных на тестовые и тренировочные, смешать их и с помощью скрипта случайным образом разделить в соотношении 80 % тренировочные данные и 20 % тестовые и рассмотреть возможность аугментации данных.

В таком случае объем данных датасета составит 27 279 экземпляров изображений и столько же аннотаций в формате InkML.

СПИСОК ЛИТЕРАТУРЫ

1. Гонсалес, Р. Ц. Цифровая обработка изображений / Р. Ц. Гонсалес, Р. Е. Вудс. – 3-е издание. – Москва : Техносфера, 2012. – 1072 с. – Текст : непосредственный.
2. Рашка, С. Python и машинное обучение / С. Рашка, В. Миржалили. – 3-е издание. – Бирмингем : Packt Publishing, 2019. – 770 с. – Текст : непосредственный.
3. Сзелиски, Р. Компьютерное зрение: алгоритмы и приложения / Р. Сзелиски. – Москва : Вильямс, 2011. – 1104 с. – Текст : непосредственный.
4. Гудфеллоу, И. Глубокое обучение : перевод с английского / И. Гудфеллоу, Й. Бенжио, А. Курвилль. – Москва : ДМК Пресс, 2018. – 732 с.
5. Бишоп, К. М. Распознавание образов и машинное обучение : перевод с английского / К. М. Бишоп. – Москва : Вильямс, 2007. – 736 с. – Текст : непосредственный.
6. Сонка, М. Обработка изображений, анализ и машинное зрение : перевод с английского / М. Сонка, В. Главач, Р. Бойл. – Москва : Вильямс, 2004. – 770 с. – Текст : непосредственный.
7. Черемисин, Д. Г. Использование нейронных сетей в задачах распознавания математических выражений / Д. Г. Черемисин, В. Р. Мкртчян. – Текст : непосредственный // Международный научный журнал «Символ науки». – 2022. – № 12 (2). – С. 34–36.
8. Карахтанов, А. А. Использование нейронных сетей для распознавания математических выражений / А. А. Карахтанов. – Текст : непосредственный // Математическое и компьютерное моделирование в экономике, страховании и управлении рисками. – № 5. –2020. – С. 86–89.
9. Шолле, Ф. Глубокое обучение с использованием Python / Ф. Шолле. – 2-е издание. – Шелтер-Айленд : Manning Publications, 2021. – 544 с. – Текст : непосредственный.
10. Комплексная платформа для машинного обучения. – Текст : элкутронный // TensorFlow. – URL: https://www.tensorflow.org/ (дата обращения: 11.06.2024).
11. OpenCV modules // OpenCV. – URL: https://docs.opencv.org/4.10.0/ (date of application: 11.06.2024).