Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики
Кафедра «Интеллектуальные системы»

Матлин Даниил Сергеевич

# Разработка и реализация алгоритмов распознавания и поиска математических формул

010990 — Интеллектуальный анализ данных

Выпускная квалификационная работа магистра

**Научный руководитель:**
д. ф.-м. н. Серебряков Владимир Алексеевич

Москва
2020

# Содержание

**Аннотация**

Данная работа посвящена разработке и реализации алгоритма информационного поиска для математических формул. Поиск по математическим формулам важен для информационных систем, созданных для работы с документами, содержащих формулы. Для решения задачи поиска по математическим формулам предлагается использование моделей полнотекстового поиска. Рассмотрены несколько существующих способов представления формулы в тексте документа,среди которых язык разметки Mathematical Markup Language, или MathML, является одним из наиболее распространенных. Он и будет являтся языком описания искомой формулы и формул-кандидатов.

**Ключевые слова**: *поиск по математическим формулам, информационный поиск, векторная модель семантики, tf-idf, MathML*

# Введение

**Актуальность темы.** Данная магистерская работа посвящена исследованию задачи поиска по математическим формулам с использованием средств полнотекстового поиска и попытке разработки и реализации соответствующего алгоритма. Решение данной задачи позволит существенно повысить качество поиска по документам, содержащим математические формулы, особенно при одновременном использовании в качестве запроса математического выражения, и ключевых слов.

Такого рода подход актуален для студентов, преподавателей и научных сотрудников при работе с публикациями или учебными пособиями, может быть использован при проверке содержания публикаций на плагиат, особенно в случае, когда два документа написаны на разных языках. Помимо использования параллельно с тектовым поиском, поиск по формулам имеет значение и сам по себе. Используя унификацию чисел, обозначений функций и переменных или рассматривая взаимное расположение элементов в формуле, можно обеспечить обнаружение семантически схожих формул, использующих различные формы записи и обозначения.

Решение задачи будет включать несколько этапов: рассмотрение доступных способов представления математических формул в цифровом виде - способы описания математических элементов при включении их в тексты публикаций, выбор наиболее подходящей модели полнотекстового поиска, разработка словаря для выбранного представления и, наконец, разработка алгоритма с последующей реализацией в рамках прототипа системы, обеспечивающей хранение набора набора формул и необходимых для поиска элементов внутреннего представления, выделенных при их обработке.

**Цель работы.** Целью работы является разработка и реализация алгоритма полнотекстового поиска и создание прототипа поисковой системы. В ходе работы решаются следующие задачи:

- исследование существующих стандартов отображения формул в цифровом виде;
- исследование применимости в данной задаче моделей полнотекстового поиска;
- разработка внутреннего представления формулы на основе выбранного стандарта представления и модели;
- разработка хранилища для набора формул иэлементов внутреннего представления;
- достижение качества результатов, сопоставимого с аналогичными системами.

**Методы исследования.** Для достижения поставленных целей используются методы полнотекстового поиска. Для программной реализации разработанного алгоритма и прототипа системы использовался язык разметки Presentation MathML, язык программирования Python 3 и модуль библиотеки встраиваемой СУБД SQLite.

**Основные положения, выносимые на защиту.**

1. Алгоритм поиска по математическим формулам, использующий векторную модель формулы

2. Экспериментальное исследование разработанного алгоритма

**Научная новизна.** Исследован подход к задаче информационного поиска по математическим формулам на основе моделей, используемых в задачах полнотекстового поиска. Предложен алгоритм поиска по математическим формулам с использованием векторной модели.

**Теоретическая значимость.** Данное исследование демонстрирует конкурентоспособность алгоритмов математического поиска, основанных на моделях полнотекстового поиска, а именно векторной модели, по сравнению с методами, использующими моделирование структуры формулы.

**Практическая значимость.** Разработанный алгоритм показывает качество работы, сопоставимое с результатами работы аналогичных систем, основанных на обрнаружении структурно схожих формул. Разработан прототип системы, реализующий алгоритм и обеспечивающий хранение корпуса формул.

**Степень достоверности и апробация работы.** Достоверность результатов подтверждена экспериментальной проверкой полученного алгоритма на реальном корпусе математических публикаций.

# Обозначения

- MathML - Mathematical Markup Language - математический язык разметки,
- $d(x, y)$ - расстояние между точками в метрическом пространстве,
- $tf$ - term frequency - отношение количества данного элемента к общему числу элементов в формуле,
- $idf$ - inverse document frequency - инверсия частоты, с которой данный элемент встречается в формулах набора,
- $tf - idf$ - произведение $tf$ и $idf$.

# 1 Постановка задачи по разработке и реализации алгоритма

В рамках данной работы требуется разработать алгоритм поиска по математическим формулам в представлении MathML. Необходимо получить программную реализацию алгоритма, спроектировать хранилище и взаимодействие алгоритма с хранимой информацией для получения прототипа системы поиска, которая может быть встроена в существующую информационную систему как дополнительный модуль поиска по формулам. Результаты поиска могут частично совпадать с искомой формулой и будут упорядочены по убыванию схожести, то есть те фломулы, что в точности совпадают с искомой, должны находиться выше остальных в списке результатов. Алгоритм должен быть реализован на языке программирования Python и в качестве хранилища использовать реляционную базу данных. Архитектура хранилища должны предполагать возможность добавление новых данных и обновление существующих. Реализация алгоритма должна быть доступна и легко встраиваема в информационные системы, ориентированные на хранение и поиск по математическим публикациям.

# 2 Обзор существующих способов представления математических формул

В разделе проведен обзор наиболее популярных сегодня способов представления математических формул в цифровых документах, а именно:

- Графическое изображение (картинка)
- Microsoft Word
- LaTeX
- MathML

В то время как первые два способа представления из приведенного списка едва ли используются в статьях, публикуемых сегодня, ранее они использовались весьма часто, о чем говорит большое количество архивных статей прошлых лет. Распознавание текста и, в частности, математических выражений на изображениях, будь то типография или рукописные тексты, является отдельной задачей, не затрагиваемой в данной работе. Способ описания формул в Microsoft Word включает в себя Microsoft Equation Editor и MathType (последний также позволяет работать с математическими записями в таких текстовых процессорах как Apple Pages, OpenOffice, LibreOffice и Google Docs). Для MathType существуют преобразователи в LaTeX. Так, например, для редактора Writer пакета LibreOffice существует долонение Writer2LaTeX, служащий как раз этой цели.

Latex и MathML являются наиболее распространенными способами представления публикаций, в том числе содержащих математические выражения. Оба формата входят в число поддерживаемых MathJax - кроссбраузерной библиотекой JavaScript, разработанной Американским Математическим Обществом и используемой такими сайтами, как Wikipedia, arXiv, MathSciNet, Coursera и другими. MathML имеет 2 версии: Content и Presentation, ориентированные на семантику и внешнее представление соответственно. Рассмотрим LateX и обе версии MathML подробнее.

## 2.1 Сравнение представлений LateX и MathML

Чтобы наглядно сравнить эти представления, предлагается рассмотреть пример формулы, описанной с их помощью. Рассмотрим общую формулу нахождения корней квадратного уравнения:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Как запись этой формулы выглядит в представлении LaTeX:

$$x = \backslash frac\{-b\backslash pm\backslash sqrt\{b\hat{}2 - 4ac\}\}\{2a\}$$

В Presentation MathML:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<mrow>
  <mi>x</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mrow>
        <mo>−</mo>
        <mi>b</mi>
        <mo>&#x00b1;</mo>
      </mrow>
      <msqrt>
        <mrow>
          <mi>b</mi>
          <msup>
            <mn>2</mn>
          </msup>
          <mo>−</mo>
          <mrow>
            <mn>4</mn>
            <mi>a</mi>
            <mi>c</mi>
          </mrow>
        </mrow>
      </msqrt>
    </mrow>
    <mrow>
      <mn>2</mn>
      <mi>a</mi>
    </mrow>
  </mfrac>
</mrow>
</math>
```

На сравнении представлений Presentation MathML и LaTeX остановимся ниже, т.к. с точки зрения подхода, который будет применён в данной работе, принципиальной разницы между двумя данными представлениями нет. Каждое в своей записи в явном виде указывает на наличие того или иного элемента в формуле, что немаловажно при подходе к содержимому формулы как к содержимому текста.

Дело обстоит иначе с Content MathML, представление формулы из примера для которого представлено ниже:

```
1
2 <math xmlns="http://www.w3.org/1998/Math/MathML">
3 <reln>
4   <eq/>
5   <ci>x</ci>
6   <apply>
7     <divide/>
8     <apply>
9       <fn><mo>&PlusMinus;</mo></fn>
10      <apply>
11        <minus/>
12        <ci>b</ci>
13      </apply>
14      <apply>
15        <root/>
16        <apply>
17          <minus/>
18          <apply>
19            <power/>
20            <ci>b</ci>
21            <cn>2</cn>
22          </apply>
23          <apply>
24            <times/>
25            <cn>4</cn>
26            <ci>a</ci>
27            <ci>c</ci>
28          </apply>
29        </apply>
30        <cn>2</cn>
31      </apply>
32    </apply>
33    <apply>
34      <times/>
35      <cn>2</cn>
36      <ci>a</ci>
37    </apply>
38  </apply>
39 </reln>
40 </math>
```

В случае с Content MathML мы видим, что такое представление наилучшим образом подходит для рассмотрения формулы в виде древовидной структуры, где в вершинах дерева располагаются операторы, а в листьях - операнды. Оператор в данном случае может быть бинарным, или принимать любое количество аргументов. В представлении Content MathML - это хорошо видно при оспиании слагаемого

$4ac$, где оператор умножения <times> указывается единожды для последовательного перемножения 3 аргументов - оператор объявляется один раз для любого количества аргументов, что интерпретируется как последовательное действия операторов на каждую из пар указанных аргументов в порядке из указания, в случае, когда оператор бинарный.

Это отличает представление Content от Presentation и LaTeX, и делает его наименее подходящим для использования моделей полнотекстового поиска, т.к. неявное указание операторов таким образом, как это было разъяснено выше, препятствует возможности их учета как составляющих элементов формулы. Помимо этого возможности преобразования других форматов в Content MathML сильно ограничена, т.к. существует значительно меньшее количество соответсвующих инструментов и качество их работы в среднем оставляет желать лучшего. Всё это вынуждает нас отказаться от Content MathML в последующем изложении.

## 2.2 LateX и Presentation MathML

Как уже было сказано выше, с точки зрения рассмотрения формулы как текста, с последующим применением методов полнотекстового поиска, между двумя этими способами представления нет существенной разницы. Преобразователи же более распространены, и, как следствие, демонстрируют более высокое качество работы, в случае перехода от LaTeX к Presentation MahtML. Примером таких инструментов может быть уже упомянутый выше MathJax или LaTeXMathML. Для преобразования MathML в LaTeX существует библиотека MathParser.

Учитывая вышесказанное, разумно использовать в качестве "общего знаменателя"для остальных представлений одно - Presentation MathML. Дополнительными аргументами в его пользу могут служить:

- MathML - это XML, что позволяет расширить нотацию и снабдить формулу метаданными, не повлияв на отображение формулы в браузере;
- MathML поддерживает элемент <maction>, позволяющий сделать формулу или её часть интерактивной - к примеру, выводить дополнительную информацию при наведении;
- Для LaTeX существует огромное количество дополнительных библиотек, и их использование в описании формулы может вызвать проблемы при её интерпетации;
- Для MathML описанной выше проблемы не существует. Язык разрабатывается Консорциумом Всемирной паутины и его инструментарий ограничен, что в нашем случае является скорее плюсом;
- В случае поисковой системы, работа с которой предполагает наличие web-интерфейса, использование MathML естественным образом соответствует возникающим к представлению требованиям.

## 2.3 Состав элементов Presentation MathML и формирование словаря

Кратко опишем все элементы, используемые Presentation MathML:

- <maction> - позволяет привязать "действие"на странице к формуле или её участку;
- <math> - корневой элемент, используемый для выделения вставки MathML. Может содержать любое количество дочерних элементов, но не может содержать другой элемент <math>;
- <menclose> - используется для изображения поверх выражения графических элементов, определяемых атрибутами;
- <merror> - используется для изображения содержимого как сообщения об ошибке;
- <mfenced> - используется для изображения разного рода скобок;
- <mfrac> - используется для изображения дробей;
- <mglyph> - используется для изображения нестандартных символов, описание которых в юникоде не представляется возможным;
- <mi> - используется для изображения идентификаторов: переменных, обозначений функций и операторов;
- <mlabeledtr> - используется для изображения выражений по левую или правую сторону от строки в таблице или матрице;
- <mmultiscripts> - используется для описания тензоров, расширяет возможности по добавлению индексов;
- <mn> - используется для изображения чисел;
- <mo> - используется для изображения операторов;
- <mpadded> - используется для изображения дополнительного пустого пространства вокруг выделенных элементов;
- <mphantom> - используется для изображения пустого пространства на местре входящих в него элементов без изменения пространственных характеристик этих элементов;
- <mroot> - используется для изображения корня с указанием степени;
- <mrow> - используется для горизонтальной группировки набора элементов;
- <ms> - используется для объявления строковой переменной, предназначенной для распознавания нотации;

- <mspace> - используется для изображения пустого пространства - "пробела размер которого определён атрибутами;
- <msqrt> - используется для изображения квадратного корня;
- <mstyle> - определяет стиль дочерних элементов;
- <msup> - используется для изображения верхнего индекса;
- <msub> - используется для изображения нижнего индекса;
- <msubsup> - используется для изображения одновременно верхнего и нижнего индекса;
- <mtable> - используется для изображения таблиц и матриц;
- <mtr> - используется для изображения строки в матрице;
- <mtd> - используется для изображения элемента (ячейки) в матрице, используется внутри элемента <mtr>;
- <mtext> - используется для добавления текста: комментариев и аннотаций;
- <mover> - используется для изображения выделения или предела над выражением;
- <munder> - используется для изображения выделения или предела под выражением;
- <munderover> - используется для изображения выделения или предела одновременно над и под выражением;

При формировании словаря не используются стилистические элементы и элементы не несущие семантической нагрузки. Из списка выше в словаре используются: <mfrac>, <mglyph>, <mmultiscripts>, <mn>, <mroot>, <msqrt>, <msup>, <msub>, <msupsub>, <mtable>, <mtr>, <mtd>, <mover>, <munder>, <munderover>. Элементы <mi> и <mo> в силу огрехов при использовании преобразователей представлений, либо из-за опечаток, допущеных человеком при описании формулы, иногда содержат элементы друг-друга. То есть под элементом <mo> бывает указана переменная и наоборот. В связи с этим сами эти элементы в словаре учитываться не будут, но будет рассмотрено их содержимое: все "вводимые с клавиатуры"операторы и операторы, записываемые в юникоде, а так же скобки и прочие элементы нотации будут учитываться как отдельные элементы словаря. Переменные будут унифицированы по регистрам: переменные-строчные буквы алфавита и переменные-заглавные будут рассматриваться как 2 различных элемента словаря. Различия между отдельными букавами, как "g"и "f"не рассматриваются.

Помимо этого, перед тем, как с помощью регулярных выражений из текста будут извлечены элементы словаря, необходимо очистить содержимое <math> от источников возможных ошибок: в качестве примера можно привести встречающиеся записи чисел "поциферно где каждая цифра в числе описана своим элементом <mn>, использование параллельно элемента <mfenced> и скобок в элементе <mo> (в данном

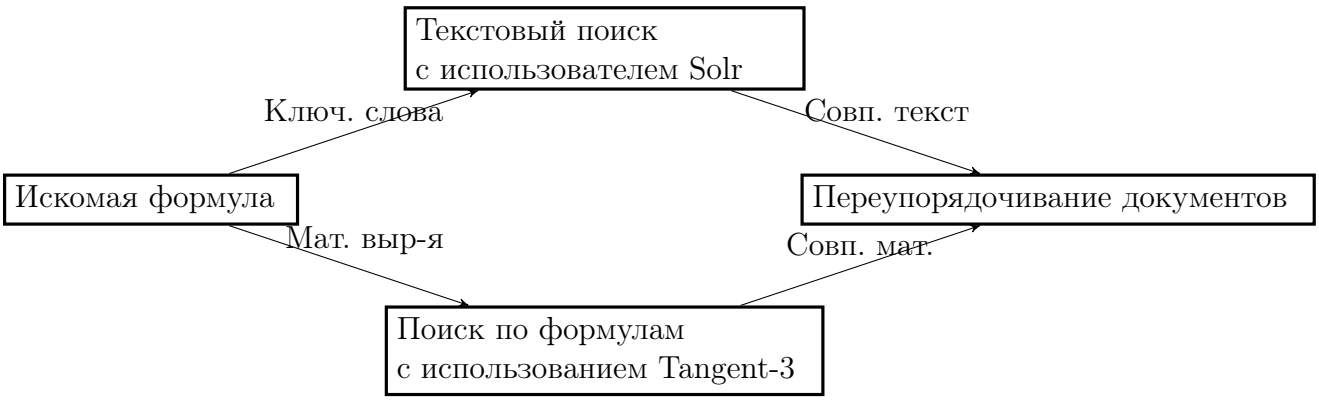
случае верным является последнее, элемент <mfenced> не используется в последних редакциях MathML), использование элементов html для описания элементов таблицы - <td> и <tr> вместо <mtd> и <mtr>, и т.п.

# 3 Обзор существующих алгоритмов поиска по математическим формулам

Наиболее заметными результатами в этой области являются решения, предложенные в рамках соревнования NTCIR MathIR - Mathematical Information Retrieval - по распознаванию математических формул, проводимого ежегодно. Среди участников и предоставляемых ими решений хочется отметить поисковый движок Tangent-3, разработанный совместно сотрудниками Рочестерского технологического института (США) и университета Ватерлоо (Канада), а так же движок полнотекстового поиска с возможностью распознавания математических формул MiaS.

Оба решения используют схожий подход: одновременное рассмотрение контекста формулы с использованием ключевых слов и поиск по математическим формулам, реализованный с помощью построения для каждой формулы дерева расположения символов, учитывающего взаимное расположение элементов в формуле, её структуру.

Модель поиска движка Tangent-3:




Предлагаемые в этих работах существенно отличаются от нашего, т.к. построение дерева предполагает учет порядка следования элементов в формуле, что позволит использовать как подход на основе n-грам, так и графовые методы. В нашем случае будет использоваться подход с использованием векторной модели и учета элементов словаря в виде "мешка слов". В данной работе нам удалось показать, что, учитывая наличие неизбежных ограничений метода, таких каких как невозможность рассмотрения вхождения формулы в формулу, результаты для поиска формул, близких по "размеру"(длине вектора без использования весов), являются конкурентными по отношению к существующим системам.

Перечисленные методы в качестве "рабочего"представления для построения деревьев используют Presentation MathML по причинам, аналогичным названным нами в предыдущем разделе.

# 4 Обзор моделей полнотекстового поиска

Кратко рассмотрим существующие подходы к моделированию текстового поиска:

- Модели, основанные на классификаторах - текст как совокупность связанных с ним атрибутов;
- Булевские модели и расширенные булевские модели - позволяют формировать запрос в виде булевского выражения, в случае расширенных моделей доступно ранжирование при частичном удовлетворении запроса;
- Векторные модели - документы и запросы являются векторами в пространстве, размерность которого соответствует размеру используемого словаря;
- Сети вывода - разновидность вероятностной модели, оценивающая вероятность удовлетворения информационных потребностей пользователя.

В данной работе будет рассматриваться подход, основанный на векторной модели, которая лучше всего соответствует способу представления формулы, предлагаемому MathML, в силу простоты построения словаря и его небольшого размера относительно размеров словарей, используемых при работе с естественным языком. Ниже будет представлен способ описания векторного пространства и последующее построение алгоритма поиска.

# 5 Описание алгоритма

Алгоритм поиска, предлагаемый в данной работе, можно условно разделить на две части: первичный отбор формул-кандидатов и последующее их упорядочение по схожести. Однако поиск подразумевает наличие подготовленных данных и хранилища, приспособленного для эффективного доступа к ним. Таким образом, описание алгоритма будет состоять из следующих частей:

1. Описание внутреннего представления и подготовка данных
2. Описание хранилища и взаимодействие с ним
3. Алгоритм поиска
   (a) Первичный отбор кандидатов
   (b) Ранжирование и выдача результатов

## 5.1 Описание внутреннего представления и подготовка данных

Векторная модель предполагает представление каждой формулы в виде точки в многомерном пространстве - вектора в векторном пространстве, в котором компонентами являются элементы сформированного нами словаря. Определим векторное пространство $X$ с заданной на нём нормой - отображением из $X$ в $\mathbb{R}$, которая соответствует длине вектора в Евклидовом пространстве и такой, что выполняются свойства:

1. $\|x\| \geq 0, \|x\| = 0 \implies x = 0$
2. $\|\lambda x\| = |\lambda| \cdot \|x\|$
3. $\|x + y\| \leq \|x\| + \|y\|$

Тогда функция расстояния $d(x, y) = |x - y|$ является метрикой и задаётся формулой

$$d(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

Таким образом, близко лежащие друг к другу точки соответствуют семантически схожим формулам.

Учитывая размер словаря и наличие большого количества формул, удобно сгруппировать вектора в матрицу, где каждая строка будет определять элемент словаря, а столбец - формулу. И формула, и запрос в данном случае будут представлены в виде набора элементов словаря, где учитывается частота их вхождения в формулу.

Пусть $\mathbf{X}$ – матрица элемент словаря - формула. В нашем случае, коллекция состоит из ∼75 тысяч формул и ∼600 уникальных элементов словаря. Матрица $\mathbf{X}$ будет иметь ∼75 тыс. строк и ∼600 столбцов. Обозначим через $w_i$ – i-ый элемени в словаре, $f_j$ – j-ую формулу в коллекции. Тогда каждый элемент $x_{ij}$ матрицы $\mathbf{X}$ – это количество вхождений элемента $w_i$ в формулу $f_j$.

Реализация алгоритма производится на языке Python 3. Для задания двухмерной структуры используется список словарей. Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов, словари - неупорядоченные коллекции произвольных объектов с доступом по ключу. Таким образом, один словарь в списке описывает одну формулу, ключами в словаре-структуре являются элементы определенного нами словаря, и элементами списка являются представленные таким образом формулы:

$$\left[ \begin{pmatrix} 'mfrac' & : & 1 \\ 'msqrt' & : & 1 \\ \vdots & & \\ 'arccos' & : & 0 \end{pmatrix}_1, \begin{pmatrix} 'mfrac' & : & 0 \\ 'msqrt' & : & 0 \\ \vdots & & \\ 'arccos' & : & 1 \end{pmatrix}_2, \begin{pmatrix} 'mfrac' & : & 0 \\ 'msqrt' & : & 0 \\ \vdots & & \\ 'arccos' & : & 0 \end{pmatrix}_3, \ldots, \begin{pmatrix} 'mfrac' & : & 3 \\ 'msqrt' & : & 1 \\ \vdots & & \\ 'arccos' & : & 0 \end{pmatrix}_n \right]$$

Идея разбиения алгоритма поиска формулы на 2 этапа строится на том, каким образом мы определям схожесть двух формул:

1) На первом этапе происходит отбор формул-кандидатов, т.е. для искомой формулы, которой в пространстве $X$ соответветсвтует точка $y$, определим множество

$$A = \{x \in X : d(x, y) < \varepsilon\},$$

состоящее из таких точек $x$, расстояние до которых удовлетворяет некоторой окрестности, в нашем случае устанавливаемой пользователем. Таким образом, мы отбираем множество формул, схожее по количеству элементов с искомой формулой. Помимо этого, ограничение числа рассматриваемых кандидатов позволяет существенно снизить вычислительную нагрузку на втором этапе алгоритма.

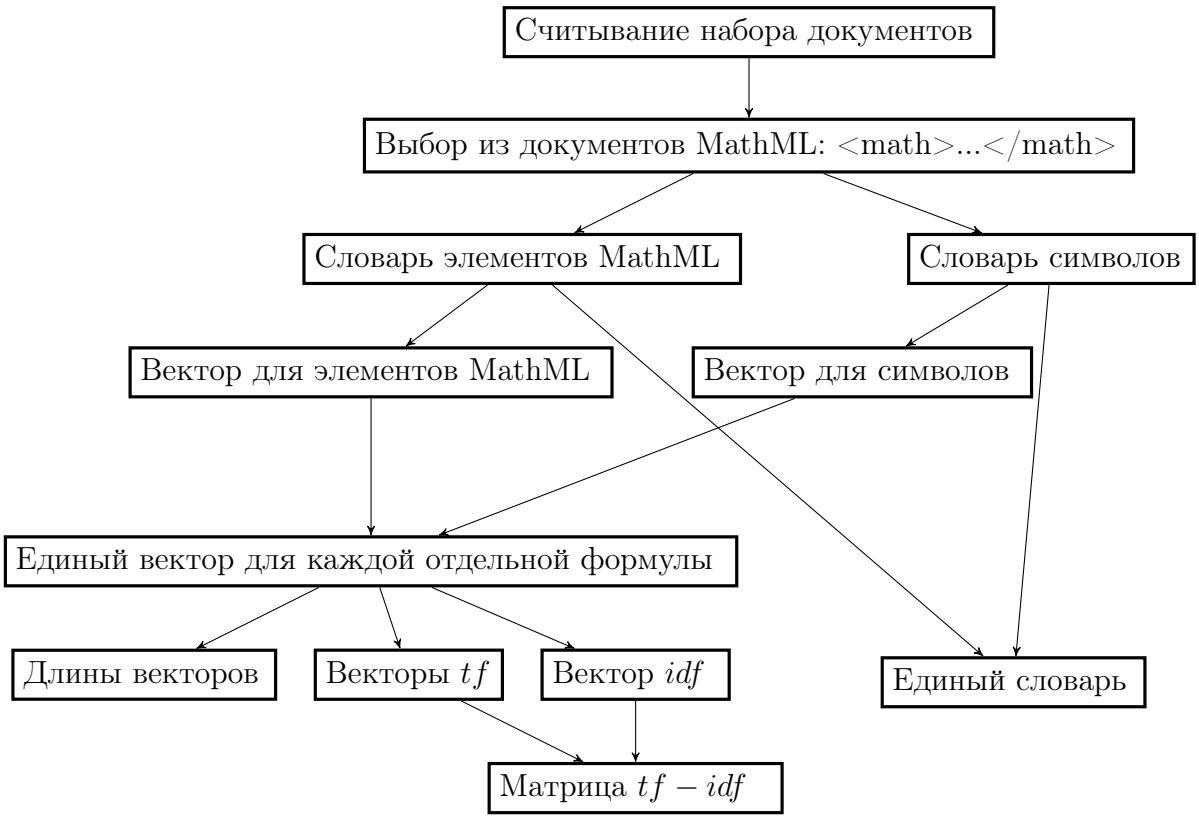
2) Второй этап требует предварительной настройки весов элементов. Идея настройки весов заключаетсяв том, что чем неожиданнее событие, тем больший интерес (и больший вес) оно представляет, и элементы словаря, встречающиеся реже, будут вносить больший вклад при определении схожести двух формул. Для взвешивания элементов нашей модели будем использовать подход $tf - idf$ (term frequency - inverse document frequency), согласно которому термин имеет большой вес, если он в некотором документе встречается часто, а в других – редко. Составляющие term frequency - отношение числа вхождений некоторого элемента словаря к общему числу элементов формулы, и inverse document frequency - инверсия частоты, с которой некоторый элемент словаря встречается в формулах коллекции, определяются следующим образом:

$$tf(t, f) = \frac{n_t}{\sum_k n_k}, \qquad idf(t, F) = ln \frac{|F|}{|\{f_i \in F | t \in f_i\}|}$$

Настройка весов помогает определить более семантически близкие формулы. Для искомой формулы вектор $tf$ рассчитывается отдельно, в качестве вектора $idf$ используется уже рассчитаный для всей коллекции вектор. Удаление наиболее употребимых элементов - а именно переменных в обоих регистрах и чисел - как стоп-слов не рассматривается, т.к. их учёт влияет на семантическую составляющую.

При подготовке данных полный состав словаря заранее не известен, точно известны только элементы Presentation MathML, которые будут учтены. Другая группа символов - содержимое элементов <mi> и <mo>: специальные символы в юникоде и символы, вводимые с клавиатуры - будет извлекаться с использованием других регулярных выражений. По этой причине предлагается для каждой группы (элементов и символов) сформировать словарь и соответсвующий вектор, чтобы затем посредством конкатенации "склеить"отдельный словари в единый словарь и вектора в единый вектор для каждой формулы.

Схема подготовки данных выглядит следующим образом:




Некоторые из полученных структур необходимо сохранять для дальнейшей обработки искомой формулы. Исходя из этапов алгоритма, которые были описаны ранее: для первого этапа потребуется сохранение словаря и длин векторов, для второго - сохранение вектора $idf$ и матрицы весов $tf - idf$. Для вывода результатов сохраняются представления формул в MathML. Далее мы рассмотрим хранилище для перечисленных данных.

## 5.2 Описание хранилища и взаимодействие с ним

Для хранения данных используется модуль библиотеки встраиваемой СУБД Sqlite для Python 3. Обращение к базе происходит посредством вызова подключения и передачи запросов. Для хранения используемой при поиске информации используются 3 таблицы: таблица для хранения формулы в представлении MathML, длины формулы. Каждая формула имеет собственный идентификатор. По нему внешним ключом с таблицей MathML связана таблица TF-IDF, в которой для соответствующей формулы хранятся веса $tf-idf$ входящих в неё элементов. В отдельной таблице хранятся словарь и вектор $idf$.

Перед загрузкой в базу данных формулы упорядочиваются по длине их вектора до учёта весов, которая и хранится в столбце Длина таблицы MathML. Таким образом, с увеличением Id формулы, увеличивается и длина её вектора, а формулы с вектором одинаковой длины расположены в таблице подряд. Это нужно для сокращения времени поиска формул в процессе первичного отбора формул-кандидатов.

Id

**MathML**

| Id | Длина | MathML |
|---|---|---|
| 1 | 1 | <math>... |
| 2 | 1 | <math>... |
| ⋮ | ⋮ | ⋮ |

**TF-IDF**

| Id | Эл1 | Эл2 | Эл3... |
|---|---|---|---|
| 1 | 1.245 | 0.0 | 0.0... |
| 2 | 0.0 | 0.0 | 2.476... |
| ⋮ | ⋮ | ⋮ | ⋮ |

**Элементы**

| EId | Элемент | IDF |
|---|---|---|
| 1 | <mfrac> | 3.03 |
| 2 | <mn> | 0.97 |
| ⋮ | ⋮ | |

Наличие связи между таблицами MathML и TF-IDF по ключу обеспечивает соответствие матрицы $tf-idf$ актуальному набору формул, представленному в таблице MathML. В случае добавления новых формул, они сохраняются в общей таблице, после чего запускается процедура расчета длин для новых формул, перерасчет матрицы $tf-idf$, т.е. в том числе и столца $idf$ таблицы Элементы. Аналогично перерасчитываются соответствующие матрица и вектор при появлении нового элемента словаря.

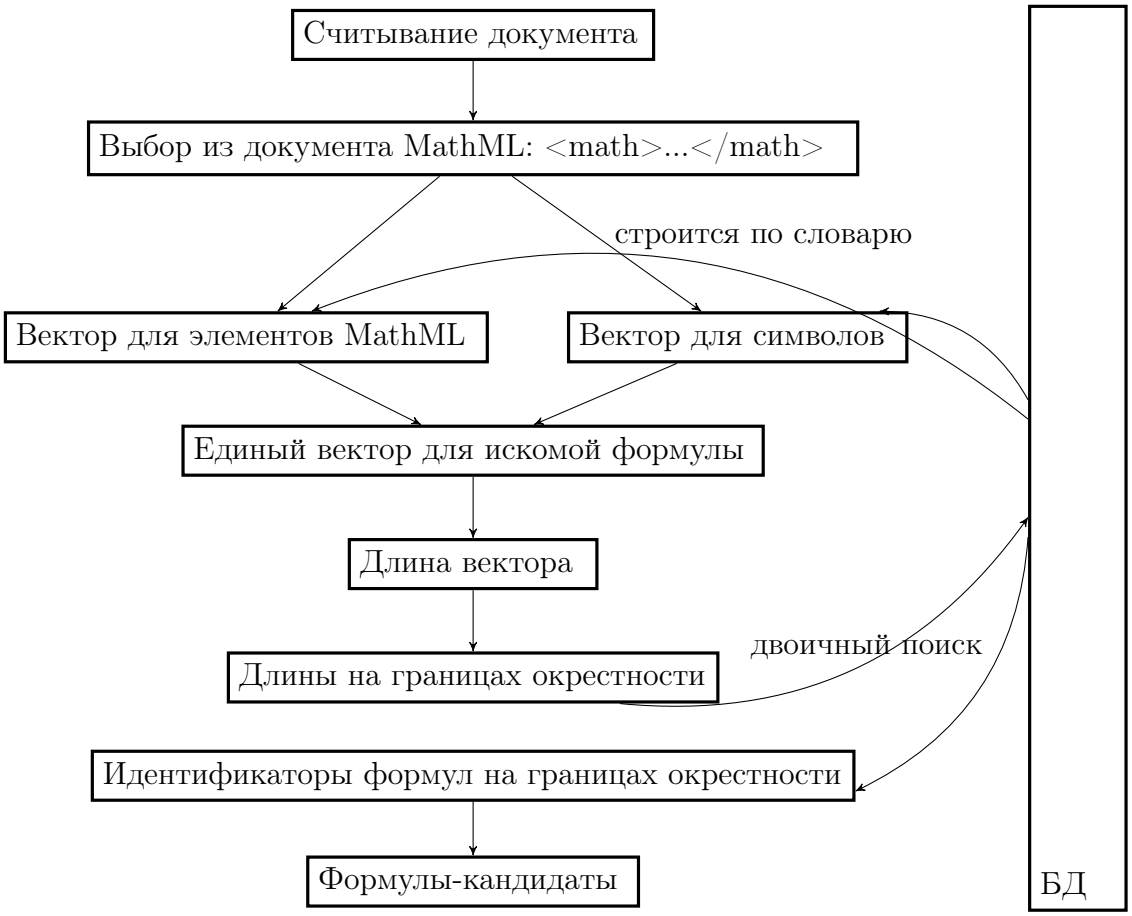
## 5.3 Алгоритм поиска

Как уже было сказано выше, поиск формул подразделяется на 2 этапа: первичный отбор кандидатов и ранжирование. Первый этап включает в себя считывание документа, содержащего MathML, извлечение формулы в MathML с последующим формированием вектора, рассчет его длины и отбор формул, попадающих в окрестность, установленную пользователем. На втором этапе производится упорядочение отобранных результатов по расстоянию между точками-формулами в пространстве элементов алфавита с применением весов $tf - idf$. Чем меньше расстояние между точками, тем семантически более близки формулы. Случай расстояния 0 считается полным совпадением, в соответствии с тем, как мы определили пространство.

### 5.3.1 Первичный отбор кандидатов

Считывание документа и извлечение регулярным выражением участка текста, содержащего MathML, происходит аналогично этапу подготовки данных. Формирование вектора также происходит аналогично этапу подготовки данных (из-за использования различных регулярных выражений), с тем отличием, что нет необходимости в формировании словаря. Далее по длине вектора происходит поиск формул-кандидатов.

Схема первого этапа алгоритма поиска:



На первом этапе работы алгоритма происходит несколько обращений к базе данных. Первое обращение происходит для извлечения словаря, который требуется для построения вектора. Второе обращение происходит при выбороре формул, длина вектора которых находится в окрестности искомой формулы, выбранной пользователем. Рассмотрим этот этап подробнее.

Извлечение всех встречающихся в наборе длин векторов формул:

```
1 SELECT DISTINCT LENGTH FROM MathML
```

Получив список всех встречающихся длин, последовательно осуществляем на нём поиск верхней и нижней границ окрестности с помощью алгоритма двоичного поиска. Хранение формул в упорядоченном по длине вектора виде позволяет использовать быстрый алгоритм поиска. Принцип работы алгоритма двоичный поиска основан на выборе элемента ровно из середины списка, с последующим сокращением рассматриваемой выборки вдвое на каждом шаге. Время его работы логарифмически зависит от входных данных, имея ассимптотическую сложность $O(\log n)$. Помимо этого, упорядоченность позволяет выбирать первое вхождение в таблицу строки с определенной длиной путём сокращения результата запроса на длину до одной строки.

```
1 SELECT Id FROM MathML WHERE Length = Border LIMIT 1
```

Т.к. формулы упорядочены по длинам, идетификаторы искомого множества формул-кандидатов будут находится в точности между идентификаторами формул-границ (для нижней границы рассматривается длина вектора на единицу большая, чтобы при выборе идетификатора на единицу большего, чем идентификатор первого её вхождения, включить все формулы с длиной, равной нижней границе).

### 5.3.2 Ранжирование и выдача результатов

После того, как были получены формулы-кандидаты, необходимо упорядочить их по возрастанию расстояния между точками в пространстве, где уже учтены веса $tf-idf$. Но сперва нужно учесть вес $tf - idf$ для искомой формулы. Вектор $tf$ вычисляется по исходному вектору, $idf$ - извлекается из базы данных:

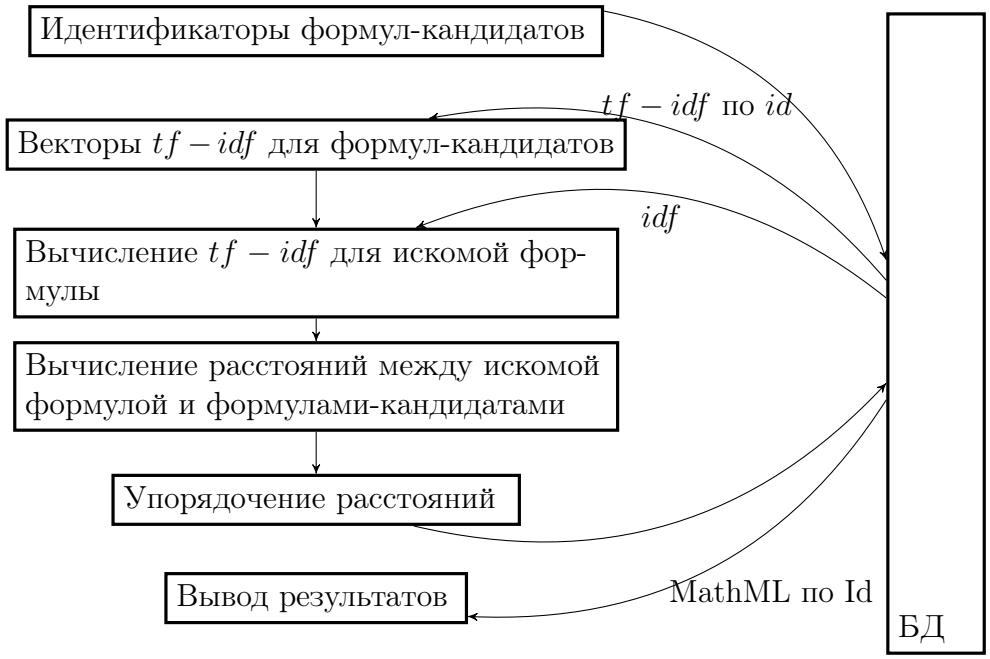
```
1 SELECT IDF FROM Elements
```

Далее поочерёдно по идентификатору формулы-кандидата из базы данных извлекается вектор $tf - idf$ формулы кандидата, после чего вычисляется расстояние между точками:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$

Когда расстояния найдены для всех формул-кандидатов, массив упорядочивается, а для интересующего количества первых формул, имеющих наименьшее расстояние

до искомой, из таблицы MathML по идентификатору извлекаются описания формул в MathML, пригодные для отображения. Происходит вывод результатов работы алгоритма. Схема второго этапа работы представлена ниже:




При выводе результатов реализована возможность отображения ссылки на статью, в которой содержится найденная формула. Выбор заголовка статьи осуществляется с помощью регулярного выражения. Возможность доступа к данной функциональности с использованием базы данных легко реализуема: достаточно добавить ещё одну таблицу, содержащую ссылки на статьи и связанную идентификатором с таблицей MathML, куда должен быть добавлен дополнительный столбец.

# 6 Демонстрация результатов

## 6.1 Результаты в случае наличия полного совпадения

**Искомая формула:**

| | | |
|---|---|---|
| $\sum_{i=1}^{k_n} a_{ni}X_{ni} \to 0$ | $\{X_n, n \geq 1\}$ | $\int_0^\infty \alpha(s,x)ds$ |

**Результаты в порядке возрастания расстояния:**

| | | |
|---|---|---|
| $\sum_{i=1}^{k_n} a_{ni}X_{ni} \to 0$ | $\{X_n, n \geq 1\}$ | $\int_0^\infty \alpha(s,x)ds$ |
| $\sum_{k=1}^{k_n} a_{nk}Z_{nk} \to 0$ | $\{U_n, n \geq 1\}$ | $\int_0^t R(t,s)b(s)ds$ |
| $\sum_{i=1}^{k_n} a_{ni}X_{ni}$ | $\{Y_n, n \geq 1\}$ | $v = \int_0^1 f^2(x)dx$ |
| $S_n = \sum_{i=1}^{k_n} a_{ni}X_{ni}$ | $\{a_n, n \geq 1\}$ | $y_n = \int_{n-1}^{n} \frac{dt}{t^{1+\alpha}}$ |
| $h = \sum_{i=1}^{n} h_i(x)e_i$ | $\{k_n, n \geq 1\}$ | $C_b^\infty(M_n, x)$ |

## 6.2 Результаты в случае частичного совпадения

**Искомая формула:**

| | | |
|---|---|---|
| $\int\limits_0^\infty f(x)\mathrm{dx}$ | $a^2 + b^2 = c^2$ | $\frac{a+\sqrt{b}}{c}$ |

**Результаты в порядке возрастания расстояния:**

| | | |
|---|---|---|
| $\int_0^\infty \alpha(s,x)ds$ | $x^p + y^p = z^p$ | $\frac{r}{\sqrt{1+r^4}}$ |
| $\int_0^t R(t,s)b(s)ds$ | $x^{r-1} + y^r = 0$ | $\frac{1}{\sqrt{2}}T$ |
| $v = \int_0^1 f^2(x)dx$ | $e^2, e^3, e^4$ | $\frac{r^*}{2\sqrt{2}} + i\frac{r^*}{2\sqrt{2}}$ |
| $Z = \int_0^{+\infty} e^{\lambda t}\varphi_{t*}Y dt$ | $R = R^0 + \mathcal{J}R^0\mathcal{J}$ | $\frac{r^*}{\sqrt{2}} + i\frac{r^*}{\sqrt{2}}$ |
| $\int\limits_\Omega M(2z(x))dx < +\infty$ | $y^3 + py + q = 0$ | $i\frac{r^*}{\sqrt{2}}$ |

**Искомая формула:** $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

**Результаты в порядке возрастания расстояния:**

1. $g = \begin{pmatrix} 0 & x & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ 2. $B = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ 3. $\left\{ \begin{pmatrix} 0 & x & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\}$

4. $\left\{ \begin{pmatrix} 0 & x & 0 \\ 0 & 0 & x \\ 0 & 0 & 0 \end{pmatrix} \right\}$ 5. $\left\{ \begin{pmatrix} x & 0 & 0 \\ 0 & 0 & x \\ 0 & 0 & 0 \end{pmatrix} \right\}$

Видно, что результаты поиска релевантны искомой формуле, в особенности для случая, когда в наборе присутствует большое количество формул, находящихся на близком расстоянии от искомой - это хорошо видно для примеров в пункте 6.1, где лучшими результатами являются формулы, полностью совпадающие с искомой, а следующие по релевантности отличаются лишь незначительно, сохраняя структуру формулы. Это позволяет нам сделать вывод, что отсутствие среди результатов столь похожих друг на друга формул является недостатком набора формул, а не используемого алгоритма.

# Заключение

Были проанализированы различные способы предствления математических формул, различные подходы к полнотекстовому поиску и поиску по формулам. На основе векторной модели был разработан алгоритм поиска по математическим формулам, использующих формат Presentation MathML.

Алгоритм был реализован в виде программы на языке Python 3. Создан прототип поисковой системы, который может быть использован для поиска по формулам в существующих информационных системах.

# Список литературы

[1] SOJKA, Petr and Martin LÍŠKA. *The Art of Mathematics Retrieval.* In Matthew R. B. Hardy, Frank Wm. Tompa. Proceedings of the 2011 ACM Symposium on Document Engineering. Mountain View, CA, USA: ACM, 2011. p. 57–60. ISBN 978-1-4503-0863-2. doi:10.1145/2034691.2034703.

[2] Kenny Davila, Richard Zanibbi, Andrew Kane and Frank Wm. Tompa. *Tangent-3 at the NTCIR-12 MathIR Task.*, 2016.

[3] Xiaoyan Lin, Liangcai Gao, Xuan Hu, Zhi Tang, Yingnan Xiao and Xiaozhong Liu. *A Mathematics Retrieval System for Formulae in Layout Presentations.*, 2014.

[4] `http://www.machinelearning.ru/wiki/index.php?title=Векторная модель`