Аннотация

Обработка текстов на естественном языке — одно из направлений математической лингвистики, изучающей проблемы компьютерного анализа и синтеза (генерации) текстов на естественных языках. Одной из подзадач данного направления является анализ математических текстов, для которой основным форматом, используемым для написания документов и научных статей, является LaTeX. В данной работе рассмотрено построение классификатора определений в научных статьях «сырых» LaTeX документов. Составлен корпус документов, содержащих определения. Исследованы различные векторные представления и их визуальное отображение для данной задачи. Исследована структура построения предложений и качество работы классификаторов на различных векторных представлениях. Наиболее высокое качество классификации было получено векторным представлением модели MathBERT с классификатором на основе градиентного бустинга.

Обозначения и сокращения

XML – eXtensible Markup Language (расширяемый язык разметки);

TF – Term Frequency (частота встречаемости термина);

NLP – Nature Language Processing (обработка естественного языка);

IDF – Inversed Document Frequency (обратная частота встречаемости в документе);

BOW – Bag of Words (мешок слов);

CBOW – Continuous Bag of Words (непрерывный мешок слов);

GloVe – Global Vectors;

USE – Universal Sentence Encoder (универсальный кодировщик предложений);

DAN – Deep Averaging Network (усредняющие глубокие нейронные сети);

BERT – Bidirectional Encoder Representations from Transformers (двунаправленное представление кодировщика трансоформера);

t-SNE – t-distributed Stochastic Neighbor Embedding (стохастическое вложение соседей с t-распределением);

NCE – Noise Contrastive Estimation (шумовая контрастная оценка);

API – Application Programming Interface (программный интерфейс приложения);

Оглавление

Введение

Обработка естественного языка - одно из популярных направлений в анализе данных. В современном мире одной из подзадач данного направления является автоматическая обработка математического текста, как языка сложной структуры и семантики.

Математический текст представляет собой сильно структурированный язык и взаимосвязь между словами и символами также не похоже ни на что из любого другого вида языка, естественного или искусственного, так как является более сложным. Также язык обладает сравнительно небольшим набором лексем, используемых повсеместно, особенно это касается символов, поскольку именно эти единицы текста часто используются в произвольном контексте. Так, например, «X» может означать: независимую переменную, неизвестную, координату на оси абсцисс и так далее. Поведение, взаимодействие и свойства такого символа могут сильно отличаться, в зависимости от природы объекта. То же самое относится и к словам. Рассмотрим это на примере слова «группа»: в зависимости от контекста оно может обозначать, как «совокупность чего-либо», так и конкретный термин, относящийся к теории групп. Анализ математических текстов интересен тем, что позволяет выделять представляющие интерес участки в математическом тексте, например: брать математические предложения, определять их синтаксическую структуру и извлекать лежащее в их основе значение в соответствующей логике [1].

Математические статьи и документы чаще всего редактируются с помощью формата LaTeX [2], который является общепринятым стандартом для технической литературы и научных текстов. На текущий момент рассмотрена общая применимость, качество работы, а также сложности и подходы при применении методов машинного обучения для анализа естественного языка при обработке математических текстов формата LaTeX на примерах некоторых задач: составления глоссария терминов; машинного перевода [3, 4].

В прошлом были попытки построения классификатора определений для LaTeX статей, но данный анализ не давал качественной оценки данному подходу, а также не являлся исчерпывающим [5].

В данной работе приведен анализ для математических LaTeX текстов и изучена подзадача построения модели, которая могла бы помочь выявить структуру определений, а также извлечь свойства объектов: задача построения модели, которая классифицирует предложения на «определение» и «не определение», при помощи методов машинного обучения, для «сырых» LaTeX документов.

Актуальность данной тематики обуславливается проблематикой обработки математических статей, построенных с помощью пакета LaTeX.

Цель данной работы заключается в построении корпуса документов и исследовании возможности классификации предложений, как «определений», из математических LaTeX статей с помощью построения различных векторных представлений с использованием градиентного бустинга и искусственных нейронных сетей, обученных на корпусе LaTeX документов.

1. Описание предметной области

Обработка текстов на естественном языке — одно из направлений математической лингвистики, изучающей проблемы компьютерного анализа и синтеза (генерации) текстов на естественных языках. Одной из задач в данном направлении является распознавание математического текста.

В рамках анализа математического текста проводились различные исследования: одним из практических применений служит помощь при автоматическом решении математических задач [6]. Но анализ математических текстов важен ещё и тем, что он может позволить выделять представляющие интерес участки в математическом тексте, например: брать математические предложения, определять их синтаксическую структуру и извлекать лежащее в их основе значение в соответствующей логике [1].

Так, например, данный анализ должен уметь извлекать все синтаксические и семантические свойства слова «группа» из определения группы, которое можно найти в любом учебнике по теории групп. Во-вторых, необходимо, чтобы каждый математический термин, обозначение или понятие извлекались из математического текста, а не становились неотъемлемой частью нашего анализа, исходя из семантического и синтаксического смысла предложения. В данном случае, это означает, что, не используя заранее заготовленные словари определений (которые могли бы быть получены с помощью ручной разметки, путем использования справочников и так далее), а используя только сами предложения, извлекались бы лексемы, которые представляют тесно взаимосвязанные языковые конструкции. Данный подход с использованием полной адаптивности значительно сложнее, чем непосредственное кодирование некоторой математики: адаптивно извлечь теорию множеств из математических текстов значительно сложнее, чем непосредственно описать ее. В-третьих, математика написана смесью обычных слов естественного языка и математических символов и их последовательностей, которые также можно назвать «словами» в математическом языке. Это очень разные по характеру лексические единицы - слова отличаются по смыслу от слов на естественном языке. Символы отличаются своим поведением от символов в искусственных языках, так как их поведение и взаимоотношения гораздо сложнее, а взаимодействие между словами и символами также не похоже ни на что из любого другого вида языка, естественного или искусственного, так как является более сложным. В-четвертых, из-за того, что теория должна описывать такой широкий спектр математических явлений, двусмысленность становится серьезной проблемой (полисемия является одной из распространенных проблем в обработке естественного языка, одним из примеров решения которой является извлечение контекстов, например,

одним и простейших алгоритмов, для решения этой задачи является извлечение n-грамм).

В настоящее время, одним из наиболее популярных инструментов для производства технической и научной документации (математических статей) является пакет для компьютерной верстки LaTeX [2]. LaTeX является стандартом для обмена и публикации научных документов. Данный пакет позволяет автоматизировать многие задачи набора текста и подготовки статей, включая набор текста на нескольких языках, нумерацию разделов и формул, перекрёстные ссылки, размещение иллюстраций и таблиц на странице, ведение библиографии и другое.

На данный момент, рассмотрена возможность парсинга документов LaTeX с помощью синтаксических деревьев, как документов сложной структуры [7, 8]. Данный подход терпит большие издержки при трансформации текста, так как теряется от 10% информации из-за сложной структуры, человеческих ошибок, заданных пользователем функций в документе и других причин, которые вызывают затруднения в работе синтаксических деревьев.

Из-за этого особенно важной, в разрезе анализа текстов, становится задача работы с «сырыми» LaTeX статьями, что позволит избежать необходимости парсинга и конвертации форматов, сохраняя большее количество информации для дальнейшей работы.

В прошлом были попытки построения классификатора определений для LaTeX статей, но данный анализ не давал качественной оценки данному подходу, а также не являлся исчерпывающим, в виду взятия определений, только выделенных авторами статей в формате LaTeX тегом «definition», так как при нем в статье могут оставаться другие определения, которые могли быть выделены иначе (например, простым текстовым выделением, таким как курсив или полужирное начертание) или же вовсе не размечены автором в силу субъективности [5]. Поэтому чаще «definition» не является единственным определением в тексте, так как наравне с ним могут вводится и другие. Также данный подход терпел большие издержки при переводе форматов, с помощью LaTeXML [9], так как не мог конвертировать отдельно взятые документы, в следствие чего некоторые документы не были распознаны вовсе.

Для задачи извлечения математических структур и связей необходимо декомпозировать данную проблему и решить подзадачу по построению модели распознавания основных конструкции (таких как определения).

В работе построен корпус документов и исследована возможность классификации определений, как основных смысловых единиц текста, для исследования структуры

математических текстов формата LaTeX, путем построения модели бинарной классификации предложений.

2. Методы

2.1. Apriori

Алгоритм для обнаружения взаимосвязей в базах данных транзакций [10]. Внутри алгоритма используется подход «снизу вверх», когда часто встречающиеся подмножества увеличиваются (расширяются) на один элемент за раз (данный шаг называется «генерация кандидатов»), после чего группы кандидатов проверяются на данных. Apriori использует два подхода к организации обхода: структуру хеш-дерева; поиск в ширину для эффективного подсчета наборов элементов-кандидатов. Алгоритм повторяет свое выполнение, пока не найдено дальнейших успешных расширений.

Классическим примером транзакций, является коллекция неупорядоченных наборов вида:

a) [молоко, хлеб, вода, лимонад];

b) [молоко, хлеб, губка];

c) [губка, хлеб, молоко];

d) [макароны, котлеты, вода].

Стоит отметить, что в данных наборах (b) идентичен набору (c) из-за их неупорядоченности.

Данный алгоритм устанавливает наличие правил в наборе данных, используя такие меры интересности как: support, confidence, lift.

Support (1.1) – Поддержка. Показатель частоты встречаемости набора X в базе транзакций.

$$\text{Support (X)} = \frac{|\{t \in T: X \in t\}|}{|T|} \qquad (1.1)$$

где $|\{t \in T: X \in t\}|$ – число транзакций t содержащих в себе набор X;

$|T|$ – общее число транзакций.

Доверие (Confidence) – частота совместной встречаемости наборов X и Y относительно набора X.

$$\text{Confidence (X, Y)} = \frac{Support(X \cup Y)}{Support(Y)} \qquad (1.2)$$

где $\text{Support}(X \cup Y)$ – вероятность того, что транзакция содержит наборы X и Y.

Лифт (Lift) – отношение вероятности совместного появления наборов X и Y к их вероятностям независимого появления в наборе

$$\text{Lift (X, Y)} = \frac{Support(X \cup Y)}{Support(Y) * Support(Y)} \qquad (1.3)$$

Данный алгоритм может быть использован для разведочного анализа текста и установления взаимосвязей в тексте [11].

2.2. Bag of words

Модель «мешка слов» - неупорядоченное представление документа в виде количества появлений каждого слова в тексте [12].

Концептуальный пример отображения текста в виде Bag of words:

a) [1, 3, 4, 8, 2, 2, 2, 0, 1, 0];

b) [1, 1, 3, 8, 0, 1, 0, 2, 1, 1].

Первая запись в (a) соответствует слову «яблоко», которое является первым словом в списке, и его значение равно «1», так как «яблоко» появляется в первом документе один раз. Вторая запись в (b) соответствует слову «лежит», которое является вторым словом в списке, и его значение равно «3», поскольку слово «лежит» встречается в первом документе трижды. Аналогично этому вторая запись в (b) отображает появление слова «лежит», которое является вторым словом в списке, и его значение равно «1», поскольку слово «лежит» встречается во 2-ом документе единожды. Значения записей в данной модели могут быть отображены, как единичными словами, так и *n*-граммами.

2.3. TF-IDF

Метод статистического анализа текста, позволяющий оценить важность термина в коллекции документов [13].

TF-IDF (2.3) основывается на двух мерах важности термина в наборе документов TF (2.1) и, соответственно, IDF (2.2).

$$\text{TF (t, doc)} = \frac{n_t}{\sum_{t' \in doc} n_{t'}} \qquad (2.1)$$

где t – термин в документе;

doc – документ принадлежащий коллекции документов D;

n_t – частота встречаемости термина в документе.

$$\text{IDF (t, D)} = \log \frac{|D|}{|\{doc \in D : t \in doc\}|} \qquad (2.2)$$

где $|D|$ – число документов в коллекции;

$|\{doc \in D : t \in doc\}|$ – число документов в коллекции, где встречается термин t.

$$\text{TF-IDF (t, doc, D)} = \text{TF (t, doc)} * \text{IDF (t, D)} \qquad (2.3)$$

Данная статистика позволяет, как классифицировать документы, так и проводить разведочный анализ, который позволяет выделить термины, как маркеры текста [14]. Вместе с применением *n*-грамм (подход позволяющий выделить контекст слова, путем применения окна шириной *n*) он является одним из широко используемых методов в NLP.

2.4. Word2Vec

Семейство нейросетевых методов, который используется для обучения векторных представлений слов, созданных по словарю корпуса документов [15, 16]. Векторное представление основывается на контекстной близости: слова, встречающиеся в тексте рядом с одинаковыми словами (а следовательно, имеющие схожий смысл), будут иметь близкие (по косинусному расстоянию) векторы. Данное семейство включает в себя две модели, использующие различные стратегии обучения: CBOW и Skip-gram.

2.4.1. Continuous bag of words

При обучении, модель непрерывного мешка слов ориентирована на решение задачи предсказания целевых слов исходя из контекста. В модели CBOW все тренировочные пары являются парами «контекст - слово», в которых входом является окно контекстных слов, а предсказанием целевое слово. Контекст содержит $2t$ слов, из которых t слов предшествуют целевому слову, а остальные t слов следуют за ним. Обозначим длину контекста как $m = 2t$. Таким образом входом системы является набор из m слов $w_{i-t}, w_{i-t+1} \dots w_{i-1}, w_{i+1} \dots w_{i+t-1}, w_{i+t}$. Не теряя общности, введем для пронумерованных слов контекста обозначения $w_1 \dots w_m$, а для целевого слова обозначение w. В данном случае, w возможно рассматривать как категориальную переменную, имеющую d возможных значений, где d является размером словаря. Целью создания векторных представлений слов является вычисление вероятности $P(w| w_1 w_2 \dots w_m)$ и максимизация произведения этих вероятностей по всем тренировочным примерам [15, 17].

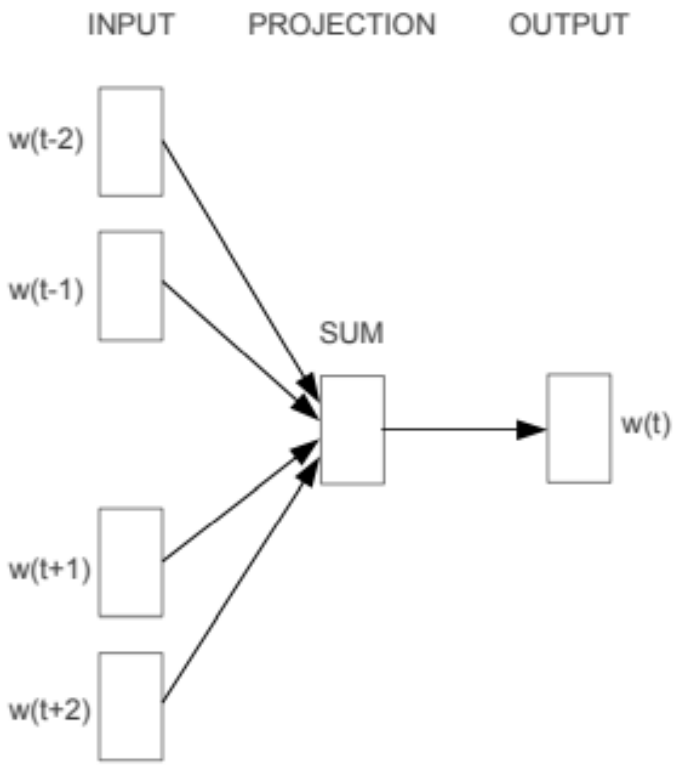

Рисунок 1 – Архитектура модели CBOW

Общая архитектура нейронной сети модели CBOW представлена на рисунке 1. В данной архитектуре присутствует один входной слой, содержащий $m \times d$ узлов, скрытый слой, содержащий p узлов, и выходной слой, содержащий d узлов. Узлы входного слоя объединены в m различных групп, каждая из которых состоит из d элементов, где каждая группа из d элементов представляет собой водной вектор для одного из m слов контекста, моделируемых в рамках модели CBOW, заданный в представлении прямого кодирования. Лишь один из этих d входов будет равен 1, в то время как все остальные входы будут равны нулю. Поэтому для входов можно использовать обозначение x_{ij}, где индексы конкретизируют позицию в контексте и идентификатор слова. Таким образом, $x_{ij} \in \{0, 1\}$, где $i \in \{1 \dots m\}$ – позиция в контексте, а $j \in \{1 \dots d\}$ – идентификатор слова.

2.4.2. Skip-gram

При обучении, модели skip-gram целевые слова используются для предсказания m слов контекста $w_{i-t}, w_{i-t+1} \dots w_{i-1}, w_{i+1} \dots w_{i+t-1}, w_{i+t}$, окружающего i-е слово в предложении. Поэтому в данном случае мы имеем один вход и m выходов. Одной из проблем при работе с моделью CBOW является то, что усреднение входных слов в окне контекста (на основе которого создается скрытое представление) приводит к благоприятному сглаживающему эффекту в случае небольшого объема данных, но не

обеспечивает в полной мере использование всех преимуществ, предоставляемых большими объемами данных [15-17].

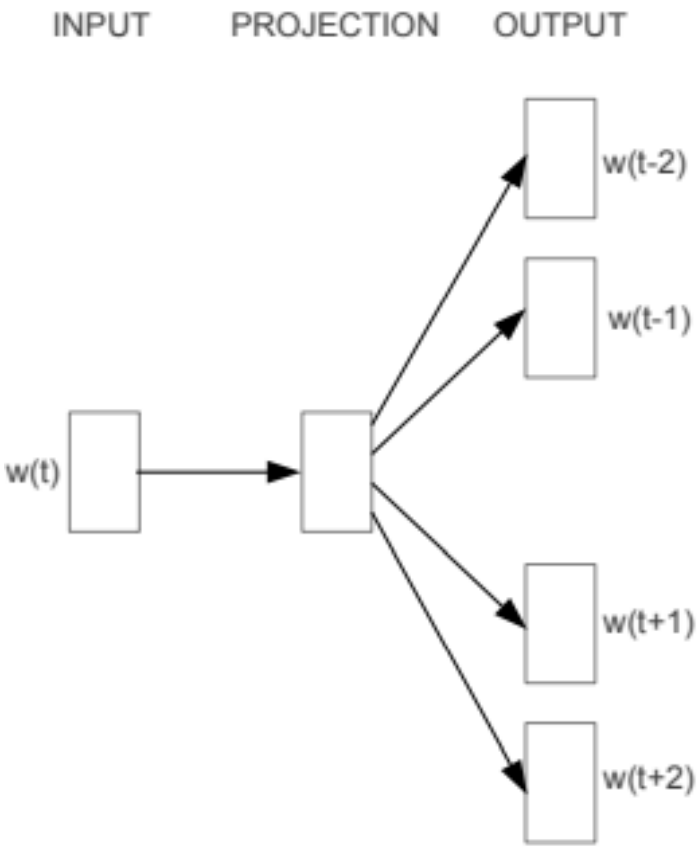

Рисунок 2 - Архитектура модели Skip-gram

В модели skip-gram входом является целевое слово w, а выходами служат m контекстных слов $w_1 \dots w_m$. Поэтому задача заключается в оценке величины $P(w_1 w_2 \dots w_m | w)$, которая отличается от величины $P(w | w_1 w_2 \dots w_m)$, оцениваемой в модели CBOW. Как и в случае модели непрерывного мешка слов, в модели skip-gram можно использовать прямое кодирование (категориального) входа и выходов. После выполнения такого кодирования модель skip-gram будет иметь бинарные входы $x_1 \dots x_d$, соответствующие d возможным значениям единственного входного слова. Аналогичным образом выход каждого тренировочного примера кодируется в виде $m \times d$ значений $y_{ij} \epsilon \{0, 1\}$ (указывает на то, принимает ли i-е слово контекста j-е возможное значение для данного тренировочного примера), где $i \epsilon \{1 \dots m\}$, а $j \epsilon \{1 \dots d\}$. Модель улучшаема с помощью негативного семплирования, NCE, иерархического softmax [16-19].

2.5. Global vectors

Алгоритм обучения без учителя для получения векторных представлений предложений. GloVe является подходом концептуально схожим с Word2Vec, но использующим в своей основе статистический подход. Ключевое отличие GloVe заключается в том, что GloVe не полагается только на близлежащие слова, а включает глобальную статистику — встречаемость слов в корпусе для получения векторов слов. Сначала строится матрица подсчетов совместной встречаемости, после чего она факторизуется и получается векторное представление. Во многих задачах векторное пространство, построенное с помощью Glove, лучше отражает закономерности в текстах, чем векторное пространство, полученное Word2ve [20].

2.6. Universal Sentence Encoder

Универсальный кодировщик предложений - модель для получения векторных представлений предложений, рассматривающая проблему переноса знаний (путем обучения на разнообразных задачах) для кодировщиков векторного представления. Данная модель опирается на два подхода к кодированию векторов предложений (документов): архитектуру кодировщика трансформера [21] и DAN [22]. Тогда как архитектура трансформера является более информационно полной и позволяющей получить большее качество обучения, DAN представляет собой подход, который уменьшает вычислительное время. Данная модель отличается от традиционных подходов, таких как GloVe и Word2Vec, тем что имеет повышенную точность работы на различных задачах [23].

2.6.1. Трансформер

Модель искусственной нейронной сети для обработки естественного языка, состоящая из двух частей кодировщика (encoder) и декодировщика (decoder), использующая механизм множественного внимания (multi-head attention). Данная модель отличается от стандартных подходов тем, что может выделять контекст и порядок слов (слева направо или справа налево) в предложении.

Механизм внимания (self-attention), иногда называемое внутренним вниманием (intra-attention), представляет собой механизм внимания, связывающий различные положения одной последовательности, чтобы вычислить представление этой последовательности. Менее формально, механизм внимания помогает создавать связи слов, но в рамках одного предложения.

Например, имеются два предложения:

а) Я клал поленья из дровницы в печку, пока она не заполнилась;

b) Я клал поленья из дровницы в печку, пока она не опустела.

В предложении (a) слово «она» связано со словом «печка», а в предложении (b) слово «она» связано со словом «дровница». Изменив слово «заполнилась» в (a) на слово «опустела» в (b), объект ссылки слова «она» изменился. В случае обработки такого предложения, знание об отношении таких слов, позволяет сохранить большее количество информации для анализа данного предложения, а также увеличить его точность в векторном представлении. Механизм множественного внимания (multi-head attention) позволяет модели совместно обрабатывать информацию из разных подпространств векторного представления в разных позициях, что было невозможно с усреднением в одном блоке механизма внимания [21].

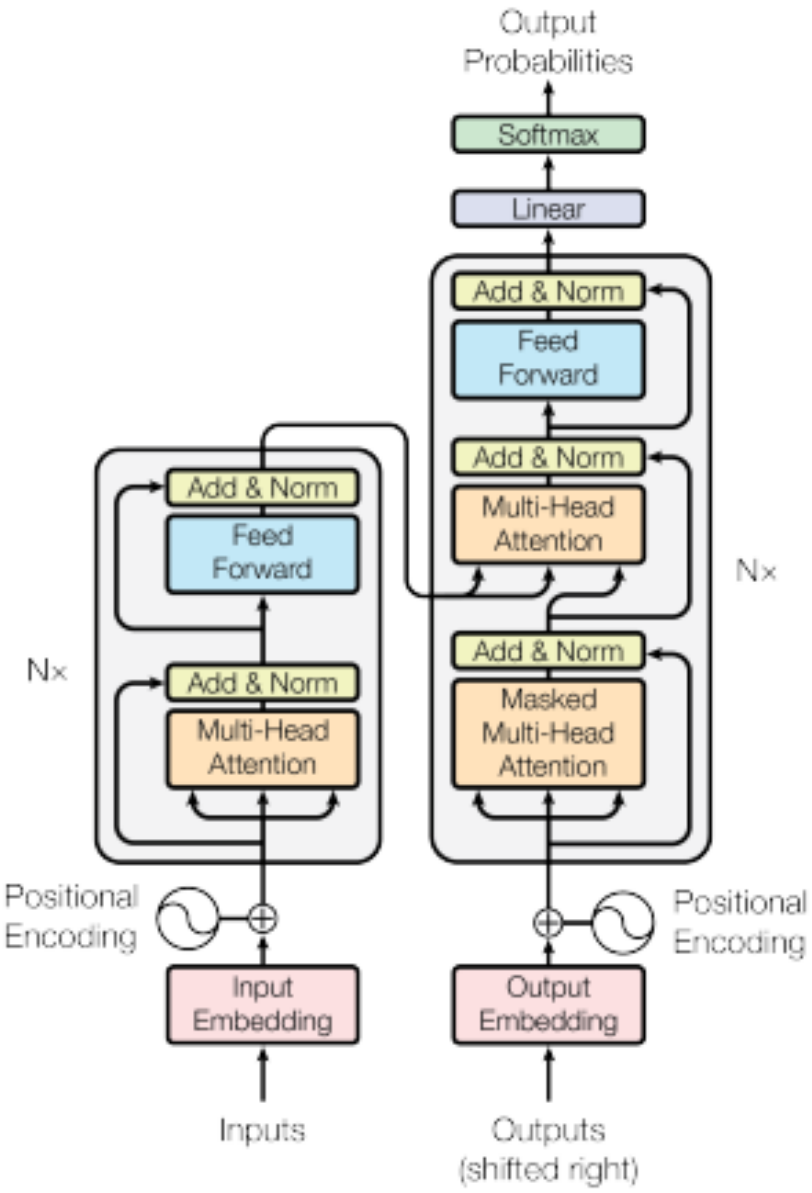

Рисунок 3 – Архитектура модели Трансформер

На рисунке 3 отображена архитектура модели Transformer: часть кодировщика (слева) и декодировщика (справа).

Кодировщик отображает входную последовательность представлений символов $(x_1 \dots x_n)$ в последовательность представлений $z = (z_1 \dots z_n)$, представленные как результат работы механизма внимания с некоторой постобработкой. Каждый кодировщик имеет два подуровня: механизм множественного внимания на входных векторах; простая полносвязная нейронная сеть с прямой связью, для проведения постобработки.

Декодировщик при заданном z генерирует выходную последовательность $(y_1 \dots y_n)$ символов по одному элементу за раз.

Механизм внимания ограничен словами, которые встречаются перед данным словом путем маскирования слов, которые встречаются после него для каждого шага: на первом шаге не маскируется только первое слово выходной последовательности, на шаге втором первые два слова не маскируются и так далее.

Каждый декодер имеет три подуровня: механизм множественного внимания с использованием масок на выходных векторах предыдущей итерации; механизм множественного внимания на выходе кодировщика и множественное внимание с использованием масок в декодировщике; простая полносвязная нейронная сеть, для проведения постобработки.

2.6.2. Deep Averaging Network

Метод композиции слов для получения векторного представления.

Для получения модели векторного представления предложений или документов, требуется соответствующая композиционная функция. Композиционная функция — это процесс объединения нескольких слов в один вектор. Композиционные функции бывают двух типов: неупорядоченная – имитирующая подход, аналогичный BOW, беря неупорядоченную статистику вложений; синтаксическая - учитывающая порядок слов и структуру предложения. Синтаксические функции (например, функция используемая в трансформере) превосходят неупорядоченные функции во многих задачах, но в то же время требуют больших вычислительных ресурсов и большее количество времени на обучение [22, 23].

DAN - неупорядоченная модель, требующая гораздо меньше времени на обучение и работающая с немного меньшей точностью по сравнению с другим подходом, например, с кодировщиком трансформера и примерно равным качеством, по сравнению с рекуррентными нейронными сетям.

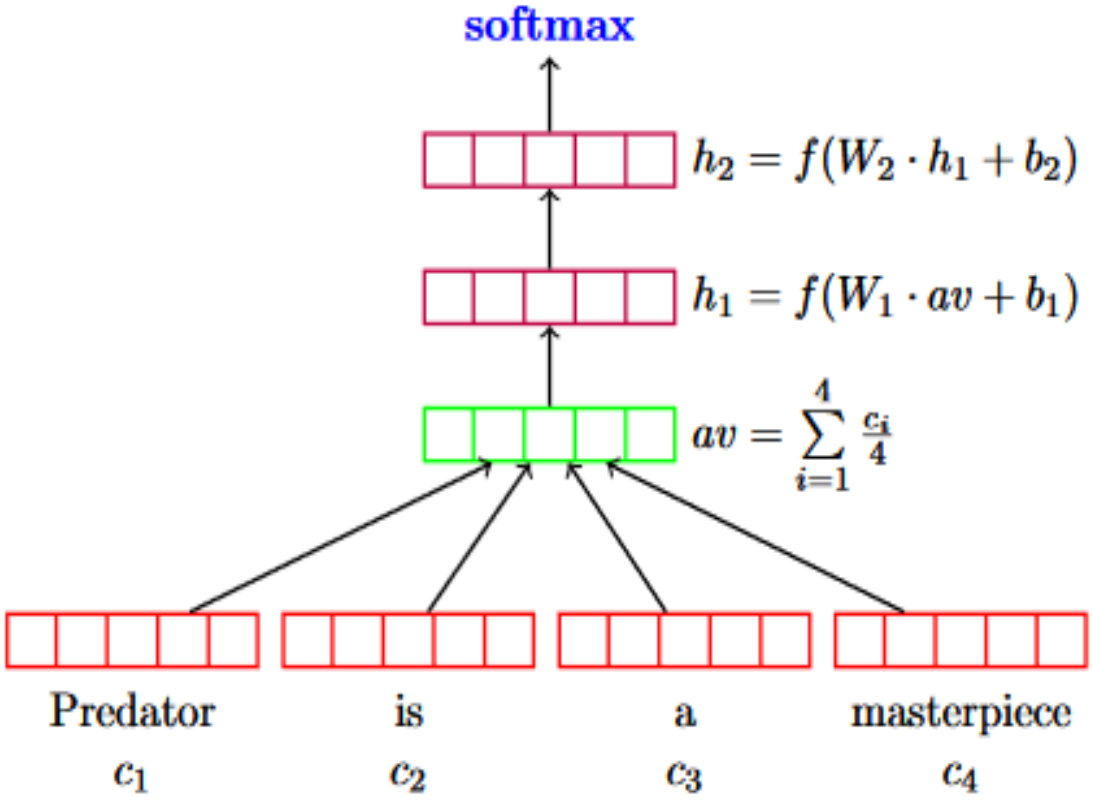

Рисунок 4 – Архитектура модели DAN

На рисунке 4 представлена архитектура модели DAN, алгоритм работы которой описывается следующими шагами:

1. Взять среднее от векторных представлений, входной последовательности;
2. Передать это среднее значение через один или несколько полносвязных слоев искусственной нейронной сети;
3. Выполнить классификацию на выходе полносвязных слоев искусственной нейронной сети.

2.7. BERT

Модель, используемая для различных классов задач в обработке естественного языка, изначально обученная на задачу маскирования текста и показывающая высокие результаты во многих задачах обработки естественного языка. Основываясь и улучшая идею кодировщика трансформера, позволяет механизму внимания [21] брать контекст как из левой, так и из правой части (в обе стороны), что позволяет улучшить результаты векторизации [24]. Внутри BERT используется только часть кодировщика из трансформера. Существует большое количество предобученных моделей данной

архитектуры, натренированных на различных корпусах текстов. В данной работе будет рассмотрено использование MathBERT [25].

MathBERT – модель архитектуры BERT натренированная на корпусе математических формул, охватывающих часть их контекста (не менее 400 символов), в том числе и LaTeX документов. MathBERT достигает высоких результатов в информационном поиске для математических формул и текста, модель была апробирована с помощью NTCIR-12 [25, 26].

2.8. t-SNE

Статистический нелинейный метод для визуализации данных большой размерности в пространстве низкой размерности (двух или трехмерном), таким образом, что близкие объекты представляются близко расположенными точками, а различные объекты, с большой вероятностью, представляются точками, далеко расположенными друг от друга [27].

t-SNE состоит из двух основных шагов:

1) Создаётся распределение вероятностей по парам объектов высокой размерности таким образом, что вероятность выбора непохожих точек будет мала, в то время как похожие объекты будут выбраны с большой вероятностью.
2) Определяется похожее распределение вероятностей по точкам в пространстве малой размерности и минимизируется расстояние Кульбака-Лейблера (4) между двумя распределениями с учётом положения точек.

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \qquad (4)$$

где P – распределение точек до снижения размерности;

Q – распределение точек после снижения размерности.

Данный метод активно используется для снижения размерности и отображения данных в обработке естественного языка [28].

2.9. Градиентный бустинг

Градиентный бустинг – методология машинного обучения по построению сильной решающей модели (ансамбля) с помощью слабых решающих моделей, чаще всего деревьев решений, для задачи обучения с учителем [29].

Алгоритм, используемый при градиентном бустинге:

1. Строятся простые модели и анализируются ошибки, совершенные моделью;

2. Определяются элементы, которые ошибочно обрабатываются простой моделью;

3. Добавляется модель, которая обрабатывает сложные случаи (где были совершены ошибки), которые были выявлены на предыдущей модели.;

4. Итеративно повторяется шаг 3;

5. Агрегируются все построенные модели, определяется вес в голосовании каждой из предсказательных моделей.

В данной работе в качестве конкретной реализации градиентного бустинга был выбран Xgboost, как один из наиболее широко используемых и зарекомендовавших себя на практике алгоритмов данного класса, получивший широкую популярность при его применении призерами в соревнованиях по машинному обучению на площадке Kaggle, получившими наилучшее качество работы моделей на данных [30, 31]

3. Инструменты.

3.1. Brat

Веб-инструмент для текстовых аннотаций, для добавления примечаний к существующим текстовым документам [32]. Brat предназначен, в частности, для структурированных аннотаций, где примечания не представляют собой текст произвольной формы, а имеют фиксированную форму, которая может быть автоматически обработана и интерпретирована компьютером.

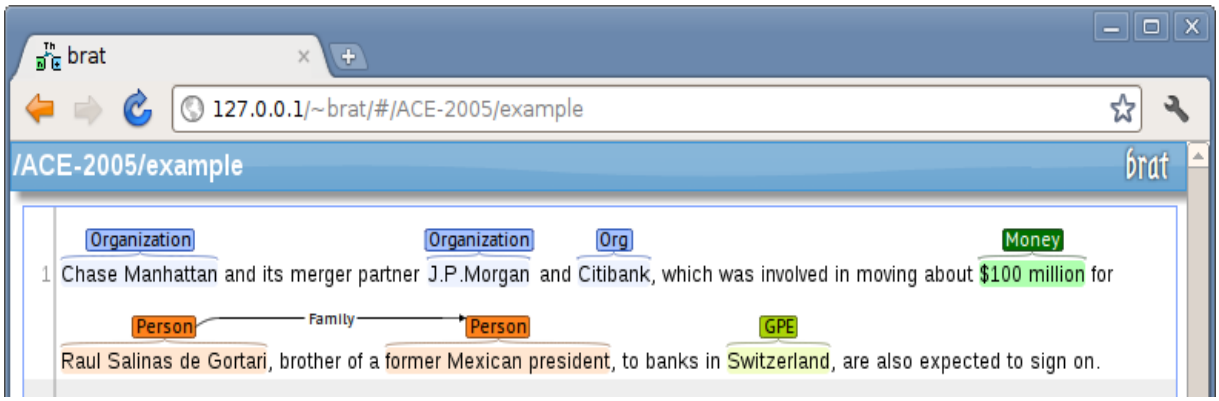

Рисунок 5 – Отображение разметки в Brat

На Рисунке 5 представлен пример, в котором предложение было аннотировано для идентификации упоминаний некоторых объектов реального мира (вещей) и их типов, а также связи между ними.

Типы аннотаций в примере:

a) Текстовые аннотации - Person и Organization;

b) Аннотации отношения – Family.

Категория простого типизированного текста подходит для создания аннотаций с целью распознавания именованных объектов и бинарных отношений для простых задач извлечения реляционной информации.

3.2. Arxiv API

Обеспечивает программный доступ к электронным статьям, размещенным на arXiv.org [33]. Данное API позволяет получить ту же информацию, которую возможно получить через HTML-интерфейс, в формате Atom 1.0. Формат Atom 1.0 определяется как грамматика XML. Данное API позволяет получить доступ к текстам статей с помощью тематического поиска или же с помощью разделения статей на категории, а также дает возможность автоматически выгрузить статьи, полученные путем запроса.

3.3. Inception

Веб-приложение, в котором несколько пользователей могут работать над одним и тем же проектом аннотаций, которое может содержать несколько проектов аннотаций одновременно [34]. Данный инструмент позволяет создать корпус, выполнив поиск во внешнем репозитории документов и добавив их. Также в приложении есть возможность использовать базы знаний, например, для таких задач, как связывание сущностей.

Рисунок 6 - Отображение разметки в Inception

На рисунке 6 представлен вид аннотаций в Inception для документа. В данном примере текст был размечен с помощью трех сущностей:

a) PER (Person) – Персона;
b) LOC (Location) – Местоположение;
c) OTH (Other) – Другое.

4. Практическая часть

Исходный код разведочного анализа, визуализации и обучения моделей, можно найти в Github репозитории[1].

4.1. Разведочный анализ

Для решения задачи обработки текста был применен разведочный анализ для извлечения из текста терминов и проверки n-грамм. Данный анализ интересен тем, что позволяет оценить практическую применимость статистических методов. Для их вычисления были использованы два подхода TF-IDF (классический подхода к анализу естественного языка) и Apriori (подход для анализа совместной встречаемости лексем, представленных в виде транзакций). Алгоритмы применялись на коллекции из 10000 документов по алгебраической геометрии. Был написан загрузчик статей с помощью Arxiv API, а также написана программа для предобработки текстовых документов.

Для получения достоверных результатов извлечения терминов данные были предобработаны с помощью выделения существительных, стемминга, а также удаления стоп-слов, служебных символов, знаков пунктуации и различных блоков графиков и картинок. В качестве тегов для блоков графиков и картинок были использованы: figure, tikzpicture, picture.

Таблица 1 – Пример результатов работы TF-IDF

№	Термин	TF-IDF
1	definable	0.472777
2	gsnc	0.470605
3	spherical	0.469460
4	tropical	0.467584
5	legendrian, curve	0.225083
6	pathfontscriptsize	0.224901
7	wt	0.224831
8	rightarrow	0.224618
9	clmg	0.224560
10	draw	0.224465
11	weakly	0.224375
12	complete, intersection, pair	0.224214
13	beginbmatrix	0.224107

[1] https://github.com/DmitriiOGU/latex_term_proj

Продолжение таблицы 1.

14	convex, geometric	0.224073
15	toroidal, variety	0.223907
16	picard, modular	0.223842
17	autorefth	0.223823
18	mathbb	0.223821
19	triideal, r	0.223711
20	smallsetminus	0.223676
21	hodge, isometry	0.223626
22	llcorner	0.221981
23	cmathbb, f	0.221920
24	mathbb	0.221714
25	cdots, cdots, cdots	0.221633
26	branching	0.221559

В таблице 1 отображен пример результатов работы TF-IDF, упорядоченный по полю «TF-IDF». Можно заметить, что среди результатов равномерно распределены специальные конструкции, принадлежащие языку LaTeX, такие как: mathbb, cmathbb, llcorner, smallsetminus и так далее. Полученные данные показывают, что математические термины и лексемы языка LaTeX, которые менее интересны для анализа (варианты начертания текста, спецсимволы), равномерно распределены по мере TF-IDF и дают сделать вывод о том, что мера TF-IDF является малоинформативной для анализа, так как n-граммы плохо извлекают основные смысловые конструкции.

Таблица 2 – Пример результатов работы Apriori

№	Транзакция	Статистика Apriori	Support
1	divisor, x	[{items base: divisor, items add: x, confidence: 0.2795860002723, lift: 3.05091941467}, {items base: x, items add: divisor, confidence: 0.0785086042065, lift: 3.05091941467}]	0.007195
2	part	[{items base: *empty*, items add: part, confidence: 0.007166486774415117, lift: 1.0}]	0.007166
3	trivial	[{items base: *empty*, items add: trivial, confidence: 0.00711392085675437, lift: 1.0}]	0.007114
4	write	[{items base: *empty*, items add: write, confidence: 0.007106912067732937, lift: 1.0}]	0.007107
5	manifold	[{items base: *empty*, items add: manifold, confidence: 0.006931692342197115, lift: 1.0}]	0.006932
6	whose	[{items base: *empty*, items add: whose, confidence: 0.006914170369643533, lift: 1.0}]	0.006914

Продолжение таблицы 2

7	bundle, line	[{items base: bundle, items add: line, confidence: 0.321469387755, lift: 17.3605637040}, {items base: line, items add: bundle, confidence: 0.3726343679031, lift: 17.36056370405}]	0.006900
8	along	[{items base: *empty*, items add: along, confidence: 0.006777498983725592, lift: 1.0}]	0.006777
9	characteristic	[{items base: *empty*, items add: characteristic, confidence: 0.006773994589214876, lift: 1.0}]	0.006774
10	notation	[{items base: *empty*, items add: notation, confidence: 0.006770490194704159, lift: 1.0}]	0.006770
11	restriction	[{items base: *empty*, items add: restriction, confidence: 0.006721428671554129, lift: 1.0}]	0.006721
12	medskip	[{items base: *empty*, items add: medskip, confidence: 0.006714419882532696, lift: 1.0}]	0.006714
13	recall	[{items base: *empty*, items add: recall, confidence: 0.006679375937425532, lift: 1.0}]	0.006679
14	complete	[{items base: *empty*, items add: complete, confidence: 0.006675871542914815, lift: 1.0}]	0.006676
15	second	[{items base: *empty*, items add: second, confidence: 0.00654270455150759, lift: 1.0}]	0.006543
16	projective, variety	[{items base: projective, items add: variety, confidence: 0.341702850670, lift: 10.62283022724}, {items base: variety, items add: projective, confidence: 0.1971892363002, lift: 10.62283022724}]	0.006343
17	curve, point	[{items base: curve, items add: point, confidence: 0.1752617293524, lift: 3.539168214641}, {items base: point, items add: curve, confidence: 0.1279456514047, lift: 3.53916821464}]	0.006336
18	unique	[{items base: *empty*, items add: unique, confidence: 0.0063219276973324546, lift: 1.0}]	0.006322

В таблице 2 отображен пример результатов работы Apriori, упорядоченный по полю «Support». Алгоритм показывает статистику неупорядоченного подхода к композиционной функции для документов (см. п. 2.1, 2.2, 2.6.2). Уровень минимального уровня расчёта меры поддержки (Support) для алгоритма 0.5%. Поле «Статистика Apriori» представляет собой кортеж из *n* элементов, где *n* – количество неупорядоченно извлеченных слов в предложении.

Расшифровка поля «Статистика Apriori»:

a) items base - показывает объект (слово), к которому присоединяется items_add, empty отображает отсутствие объекта;

b) items add - показывает объект (слово), которое присоединяется items_base;

c) confidence – условная вероятность добавления items_add к items_add;

d) lift – отношения зависимости элементов внутри поля «Транзакция» к их независимости (см. 2.1).

Итоги работы алгоритма показывают, что математические термины достаточно хорошо извлекаются статистическим подходом, так как дают возможность увидеть осмысленные математические термины (а также слова относящиеся непосредственно к тексту, а не разметке LaTeX) такие как: manifold, projective, variety и другие.

Результаты работы этих двух методов показывают общую разреженность текста (осмысленные слова и словосочетания плохо выделяются, оставляя только зарезервированные слова языка LaTeX, которые часто встречаемы), а также менее точную работу классического подхода из-за редкости терминов, которые встречаются в тексте математических статей, из-за их природы.

4.2. Подготовка данных

Для подготовки к анализу статей, написанных с помощью пакета LaTeX, была произведена разметка статей. В качестве инструмента разметки был выбран Brat. Размечаемые статьи были выгружены с сайта arxiv.org по тематике комбинаторика.

В качестве альтернативы была рассмотрена подготовка такого корпуса из LaTeX документов pdf формата (компилированного с помощью инструмента pdfTeX [35], который включен в большинство современных сборок LaTeX). Для разметки документов был использован Inception. Недостатком выбранного инструмента является то, что при не выделяются сущности: в представлении pdfTeX сущности отображены единым (не разграниченным пробелами) текстом, но для разметки сущностей они должны быть разделены между собой.

Основные сущности, которые использовались в разметке, на определения.

1. Symbol – сущность, которая описывает символом(-ами) объявленный объект в предложении.
2. Obj – сущность, описывающая определяемое понятие словом(-ами), не включая в себя описательную часть объекта, такие как: свойства, объект на котором он вводится и тому подобное.

Данные сущности были использованы для контроля ошибок, а также для продолжения работы в рамках задачи извлечения сущностей.

Правила, используемые для разметки:

1. Не выделяются сущности в предложениях в которых ничего не вводится, или дана конкретная реализация или пример объекта.
2. При разметке не выделяются сущности, которые описаны в предложениях с подсписками, таблицами и так далее, так как сложно описать их структуру линейно (существуют сложные взаимосвязи в объявлении сущности).
3. В случае, когда существует двойственность в обозначении сущности, выбирается, то, что более емко его описывает.

4.2.1. Описание разметки и примеры:

В данном разделе представлены примеры разметки для документов, для понимания общего вида предложений, их структуры, а также самой разметки.

Пример 1.

«Given a positive integer $n \in \mathbb{Z}_{+}$, we use $\mathfrak{S}_{n}$ to denote the \emph{symmetric group} on the set $[n] = \{ 1, 2, \ldots, n \}$»

В примере 1 «\mathfrak{S}_{n}» – Symbol (символьное описание объекта), а «symmetric group» – Obj (объект, описываемый в предложении). В данном примере показано, что сущность Obj может быть описана несколькими словами, также сущность Symbol может быть также описана несколькими символами.

Пример 2.

«Let $f_j:G-v_j \rightarrow G'-v'_j$ be an isomorphism for each $j\in\{1,\dots,n\}$.»

В примере 2 «f_j:G-v_j \rightarrow G'-v'_j» – Symbol (символьное описание объекта), а «isomorphism» – Obj (объект, описываемый в предложении). Данный пример отображает, что сущности типа Obj не обязательно должны отображаться выделением текста или вариантами начертания.

Пример 3.

«Furthermore, the set\[Av(\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(k)})= \bigcup_{n \geq 1} S_{n}(\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(k)})\] is called the \emph{(pattern) avoidance class}with \emph{basis} $\{\pi^{(1)}, \pi^{(2)}, \ldots,\pi^{(k)}\}$.»

В примере 3 «Av(\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(k)})= \bigcup_{n \geq 1} S_{n}(\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(k)})» – Symbol (символьное описание объекта), а «(pattern) avoidance class» – Obj (объект, описываемый в предложении). Данный пример описывает ситуацию, когда сущность Symbol не обязательно ограничена символами «$».

Пример 4.

«This motivates a bijective extension of Schensted Insertion called the \emph{RSK Correspondence}, which originated from the study of representations of the symmetric group by Robinson \cite{refRobinson1938} in1938.»

В примере 4 «RSK Correspondence» – Obj (объект, описываемый в предложении) показывает, что предложение, представляющее собой определение, может не содержать сущности типа Symbol в определении.

Пример 5.

«Given shadowlines $L^{(m)}_{i}(\sigma),L^{(m)}_{j}(\sigma)\in D_{SW}^{(m)}(\sigma)$ with$i<j$, we call $L^{(m)}_{i}(\sigma)$ the \emph{lower}shadowline and $L^{(m)}_{j}(\sigma)$ the \emph{upper}shadowline.»

В примере 5 отображено 4 объекта, что демонстрирует ситуацию, где в предложении может быть более одного определяемого объекта и символьного обозначения:

a) «L^{(m)}_{i}(\sigma)» – Symbol, символьное описание объекта 1;
b) «L^{(m)}_{j}(\sigma)» – Symbol, символьное описание объекта 2;
c) «lower}shadowline» – Obj, объект 1 описываемый в предложении;

d) «upper}shadowline» – Obj, объект 2 описываемый в предложении.

Пример 6.

«A \textbf{Steiner triple system} of order n (or $STS(n)$) is a pair (X,S) where X, called the foundation, is a finite set with n elements and S is a subset of $\mathcal{P}_3(X)$ (the set of all 3-subsets of X) whose elements are called blocks, such thatfor every pair of distinct elements of X there is exactly one block in S which contains both of them.»

В примере 6 «STS(n)» – Symbol (символьное описание объекта), а «Steiner triple system» – Obj (объект, описываемый в предложении). Данный пример описывает ситуацию, когда определение задается цепью определений, то выделяется только самый основной объект в предложении.

4.3. Визуализация векторных представлений

В данном разделе представлена визуализация результатов работы алгоритма снижения размерности t-SNE для векторных представлений набора данных, составленных с помощью моделей: BOW, TF-IDF, Word2Vec, GloVe, USE, BERT и MathBERT. Точки, отображенные на рисунках синим цветом, представляют собой предложения размеченные как определения в математических текстах. Синим цветом на рисунках отображены точки, отмеченные как класс 1-ый (целевая метка класса), красным цветом как 0-ой класс (нецелевая метка класса).

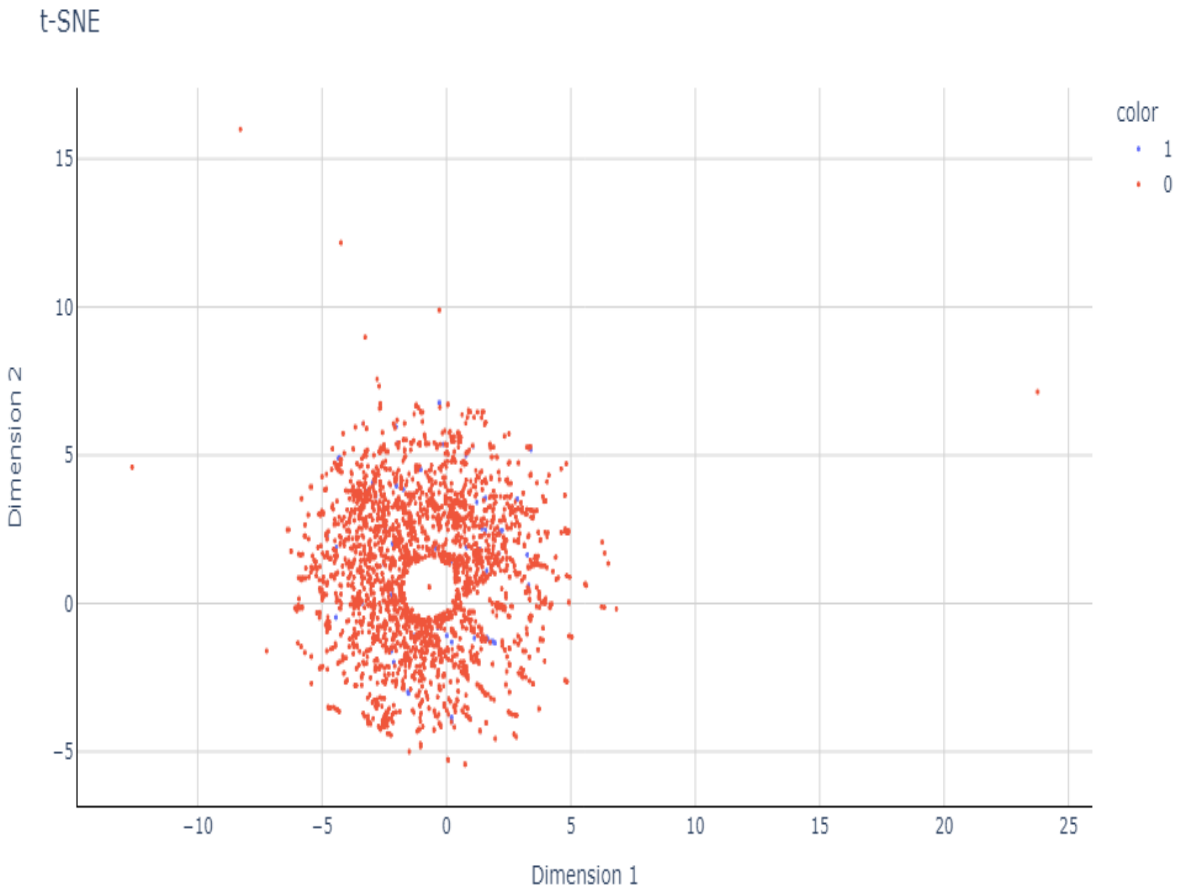

Рисунок 7. Результаты работы t-SNE на векторных представлениях модели BOW

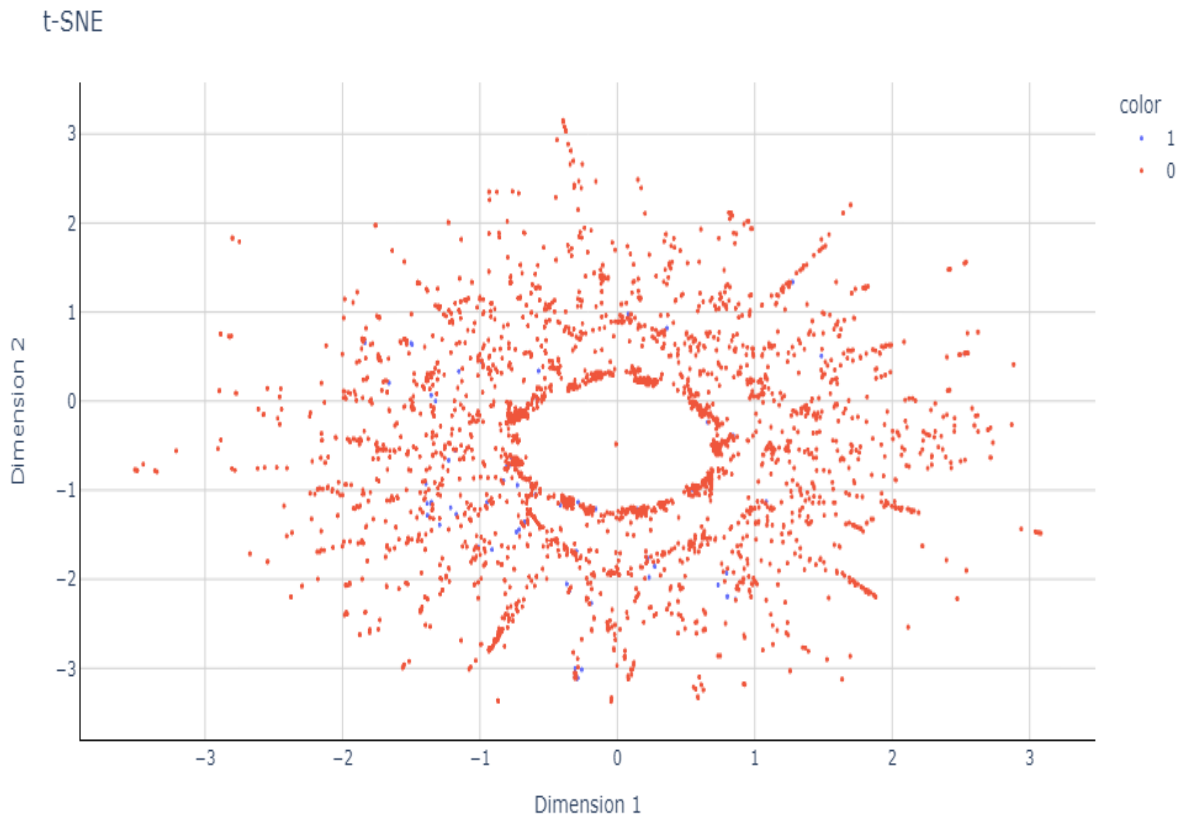

Рисунок 8. Результаты работы t-SNE на векторных представлениях модели TF-IDF:

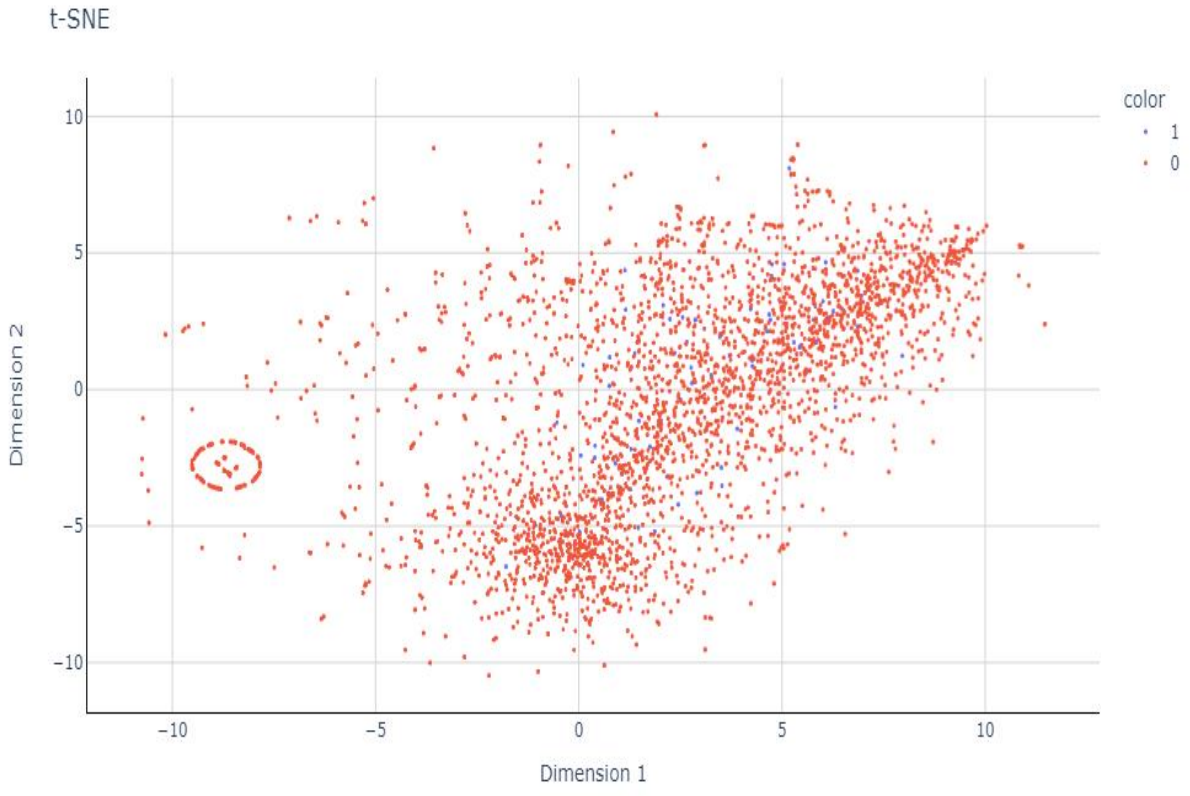

Рисунок 9. Результаты работы t-SNE на векторных представлениях модели Word2Vec.

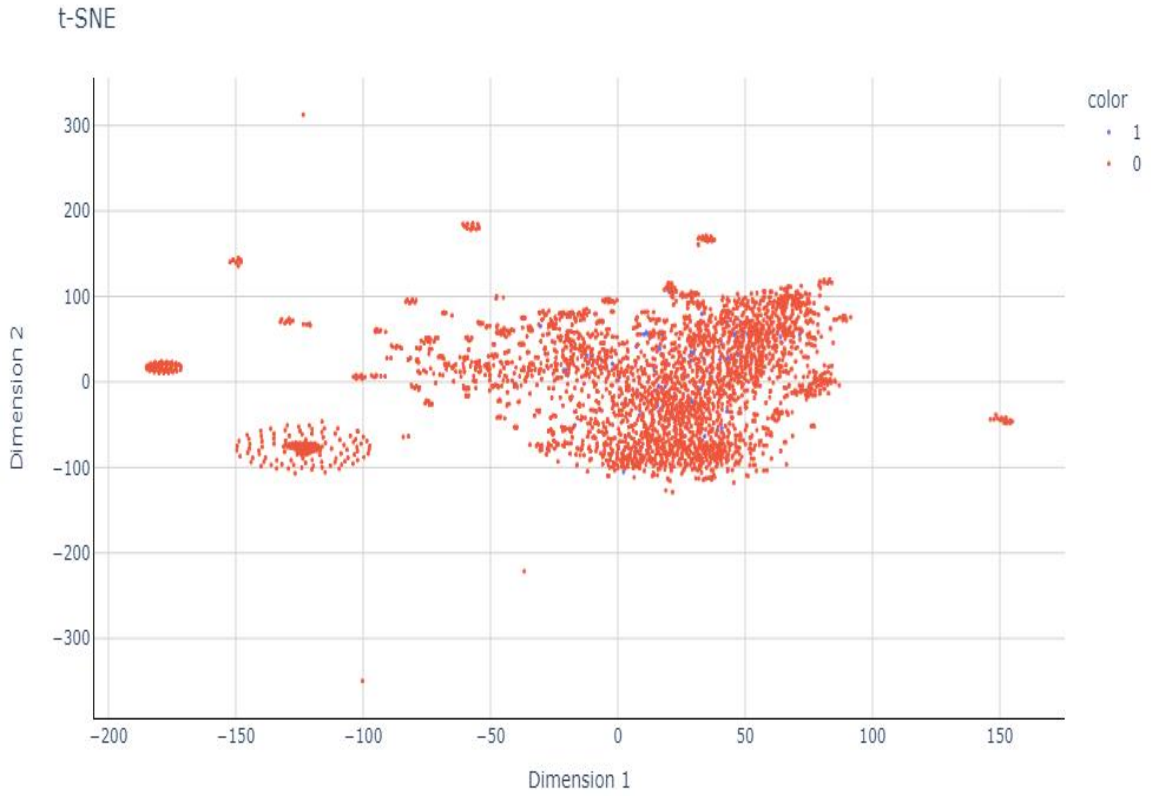

Рисунок 10. Результаты работы t-SNE на векторных представлениях модели GloVe.

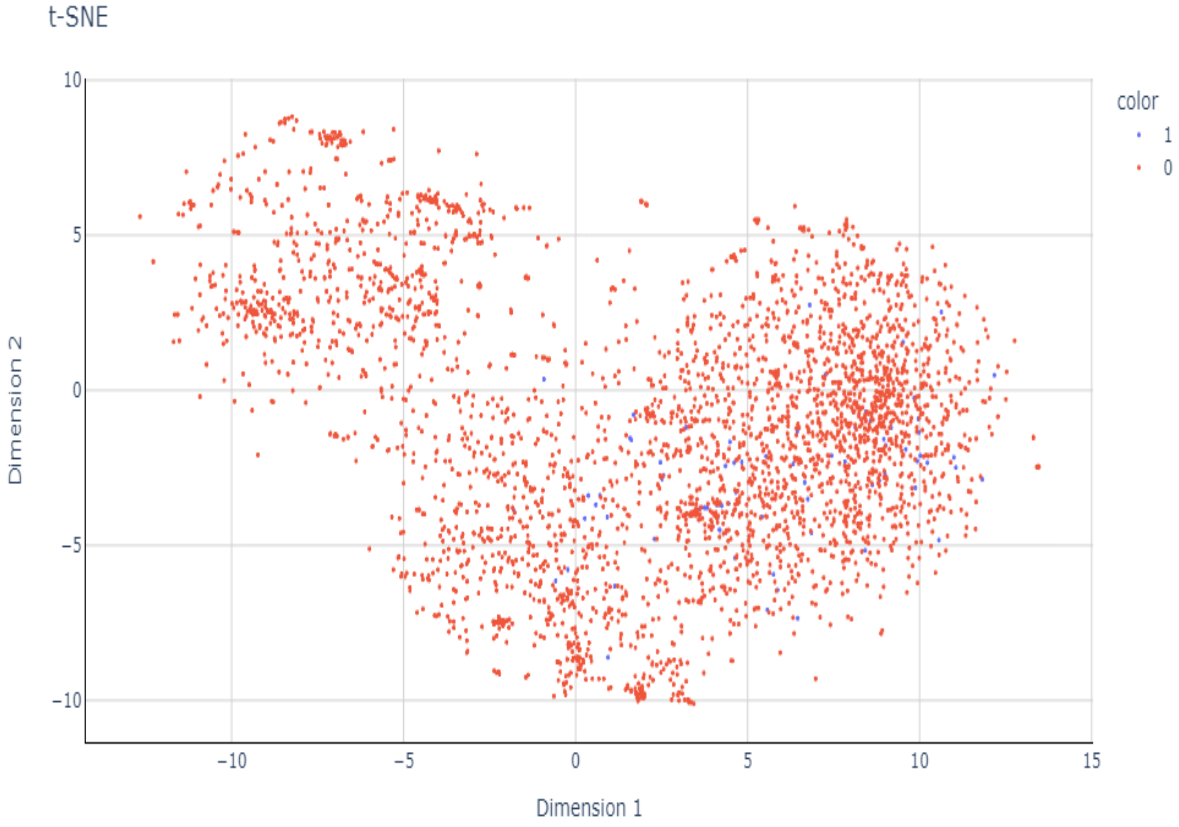

Рисунок 11. Результаты работы t-SNE на векторных представлениях модели USE

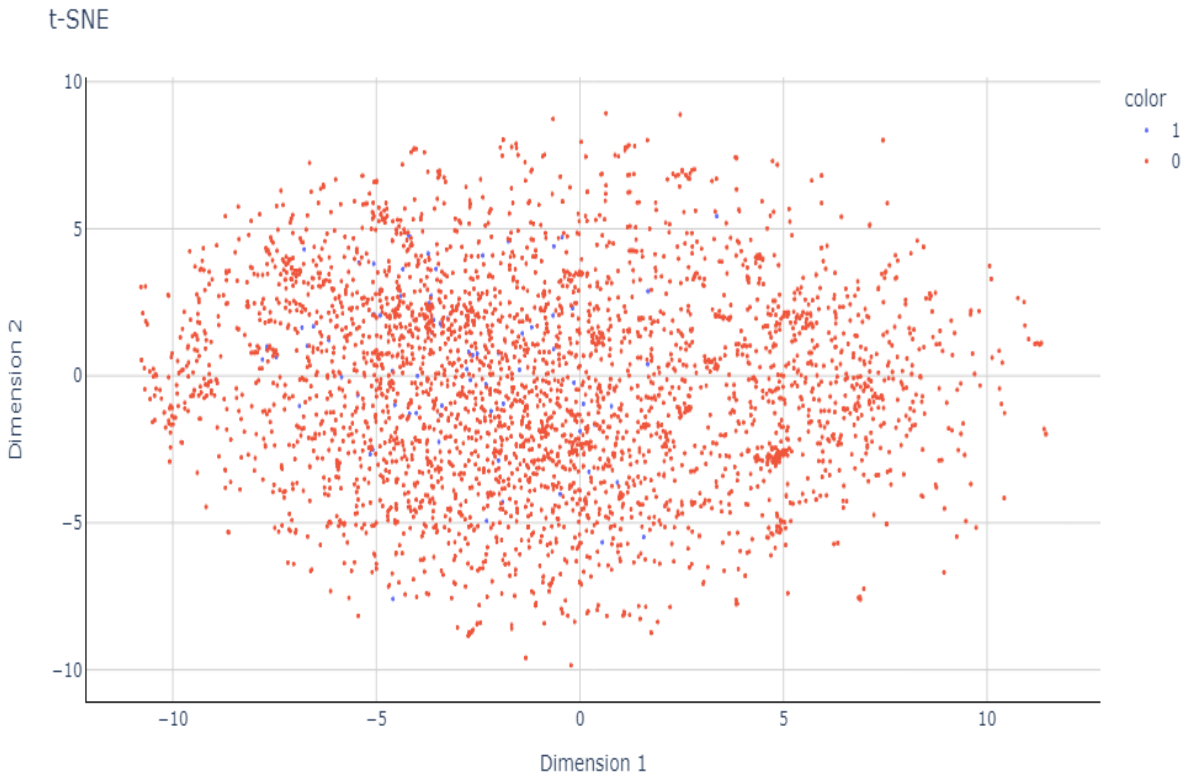

Рисунок 12. Результаты работы t-SNE на векторных представлениях модели BERT

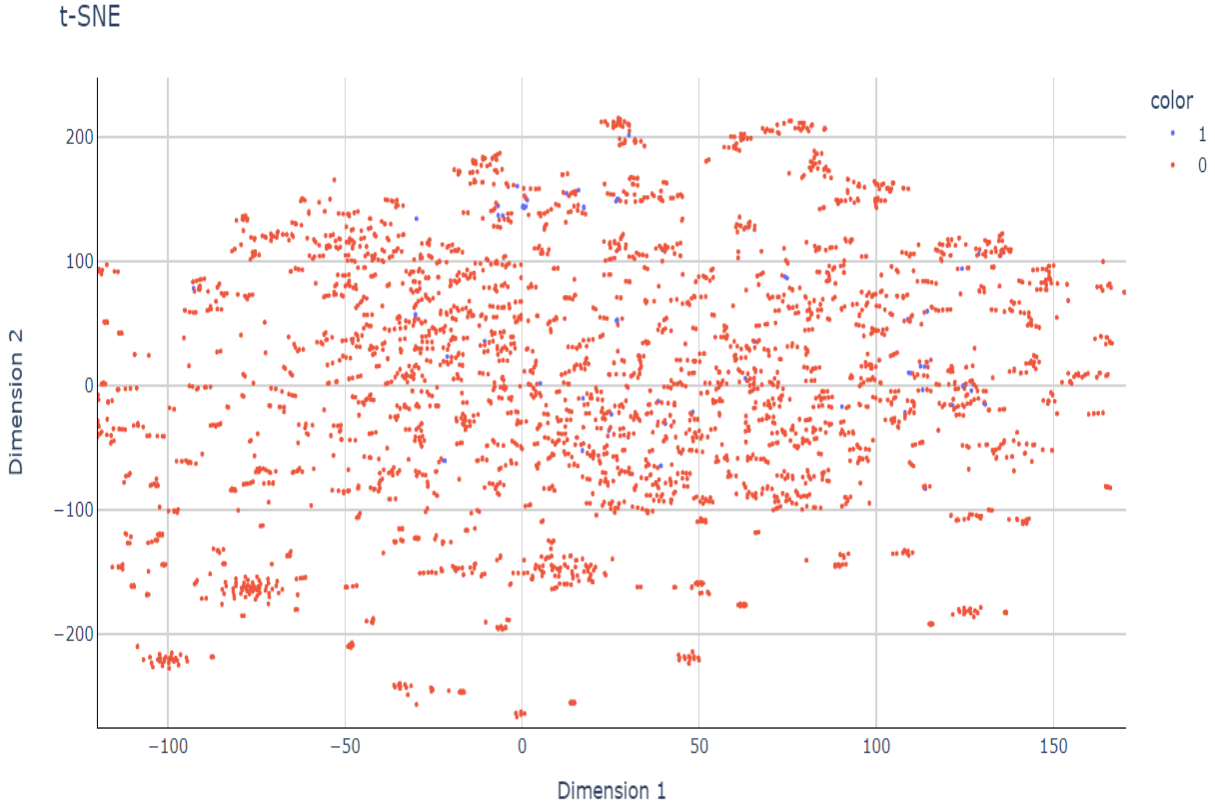

Рисунок 13. Результаты работы t-SNE на векторных представлениях модели MathBERT

На рисунках 7-13 представлены результаты работы t-SNE на векторных представлениях. Данные рисунки показывают, что метки классов четко не разделяются ни одним из предложенных векторных представлений, что делает затруднительным построение модели с хорошей обобщающей способностью.

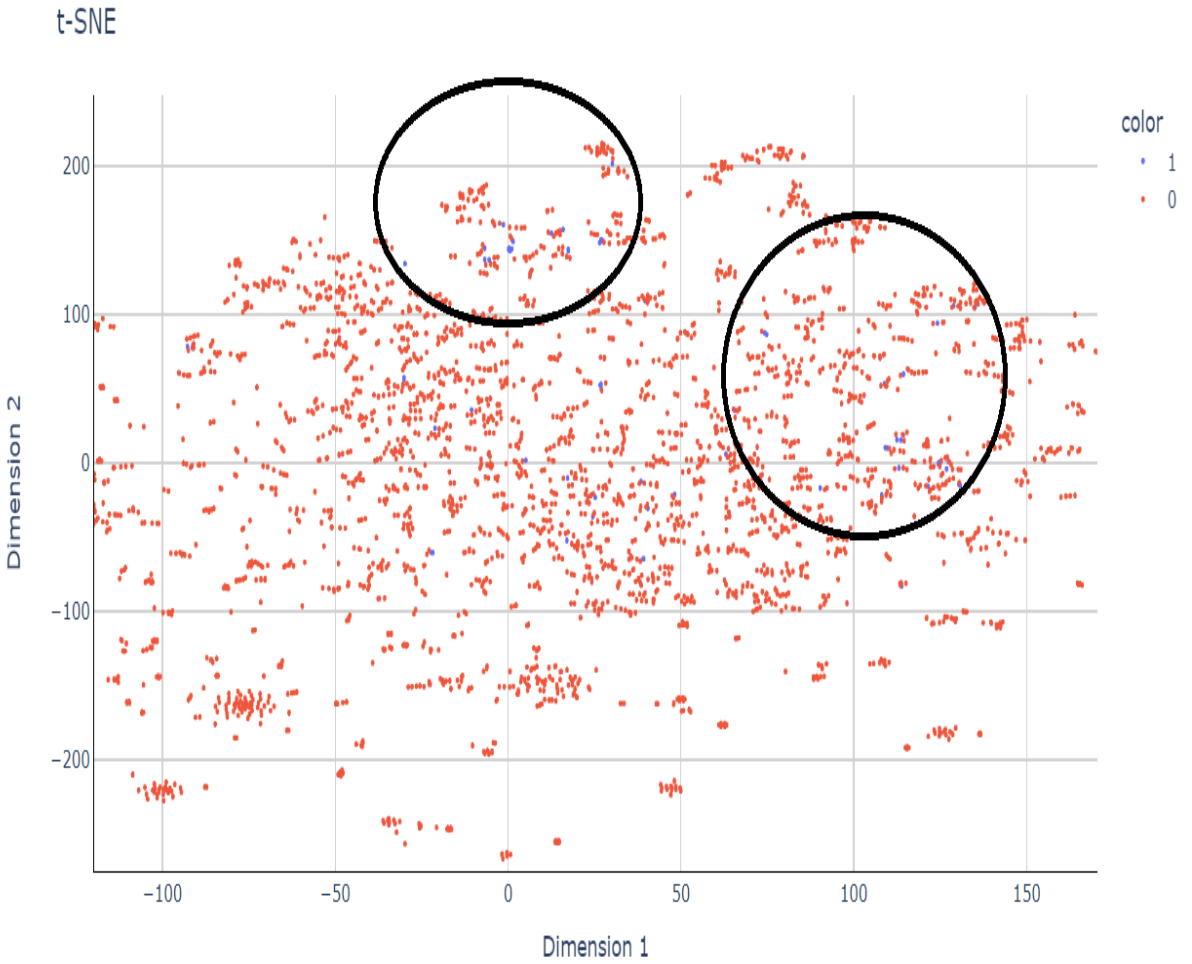

Рисунок 14 – Пример концентрации точек векторного представления, полученных с помощью модели MathBERT

На рисунке 14 можно увидеть, что выделенные черными кругами места представляют собой наибольшие скопления точек, полученные с помощью применения t-SNE к векторному представлению модели MathBERT, относительно других моделей. Данные кластера, полученные с помощью t-SNE, показывают, что точки, находящиеся рядом друг с другом, являются более близкими в изначальном векторном пространстве. Чем лучше выделяются кластера, тем лучше можно построить модель, разделяющую данные классы. Этот факт объясняется высокой близостью в векторном пространстве (схожестью объектов).

Так, например, можно увидеть, что точки представленные на рисунке 9 и рисунке 10 достаточно сильно отдалены на графике, что может дать низкие результаты при классификации объектов, то есть предложения, представляющие собой определения, для этих векторных представлений плохо отделимы от остальных предложений.

4.4. Описание проведенных экспериментов

В данном разделе обсуждаются методы и подходы для классификации определений.

Предобработка перед построением векторных представлений состояла из трех шагов:

1. Документы предварительно очищались от служебных слов, которые стояли в отдельных строчках, будучи отделенными от текста предложения (например переносом каретки) и не вносящими в него контекстуального смысла;
2. Удалялись служебные символы (по типу «\\», «{», «}» и т.д.);
3. Отделялись служебные символы («$», «\[», «\]»), чтобы внести дополнительный контекст.

После предобработки, данные, были векторизованы посредством одной из следующих моделей:

a) BOW;
b) TF-IDF;
c) Word2Vec;
d) GloVe
e) USE;
f) BERT;
g) MathBERT.

Для моделей BOW и TF-IDF представления были обучены, на корпусе предложений, составляющим около 80 тысяч единиц.

Количество данных после разметки составило 3991 предложений. Распределение меток по классам составило: 77 меток 1-го (целевого) класса и 3914 меток 0-го (не целевого) класса.

В виду несбалансированности классов в ходе работы была применены взвешенная функция потерь, которая сильнее штрафует за ошибку на определенном классе. Для борьбы с недостатком данных были использованы: перенос знаний и адаптация домена [17, 28].

Оценка качества работы алгоритма производилась с помощью:

a) Стратифицированной перекрестной проверки [17]
b) Оценки качества по F1-мере [28]

Для предсказания использовался градиентный бустинг и полносвязная нейронная сеть (для модели MathBERT). Подбор гиперпараметров моделей был оптимизирован с помощью фреймворка Optuna [36], предназначенного для их автоматического поиска.

4.4.1. Перенос знаний и адаптация домена

Термины «перенос обучения» (transfer learning) и «адаптация домена» (domain adoptation) относятся к ситуации, когда модель, обученная на одной задаче, используется для улучшения обобщаемости в другой задаче [28].

Введем два распределения: P_1, P_2. При переносе знаний обучаемая модель должна быть обучена на двух или более задачах, но предполагается, что многие факторы, объясняющие вариативность P_1, относятся и к факторам, которые следует учесть для обучения P_2. Обычно это интерпретируется в контексте обучения с учителем, когда вход один и тот же, а природа меток может быть разной. Например, можно обучиться чему-то относящемуся к одному набору зрительных объектов, скажем собакам и кошкам, а затем перейти к другим объектам, скажем бабочкам и гусеницам. Если в первом случае данных, которые были выбраны из распределения P_1, намного больше, то, возможно, следует обучить представления, полезные для быстрого обобщения при наличии лишь небольшой выборки из P_2. У многих зрительных объектов есть общие особенности, низкоуровневые признаки: границы и формы, эффекты от применения геометрических преобразований, изменений освещения и так далее. В общем случае перенос обучения, многозадачное обучение и адаптацию домена можно реализовать путем обучения представления, если существуют признаки, полезные в различных ситуациях или задачах, которые соответствуют объясняющим факторам, встречающимся в нескольких ситуациях [28].

Адаптация домена – задача, направленная на использование в новой задаче предыдущего накопленного моделью опыта; выделение некоторых характерных особенностей из домена-источника для их использования в целевом домене. Например, рассмотрим задачу определения эмоциональной тональности текста, когда нужно проанализировать, выражает ли комментарий положительные или отрицательные эмоции. Комментарии могут иметь разное происхождение. Проблема адаптации домена возникает, когда предсказательная модель эмоциональной окраски, обученная на отзывах пользователей книг, журналов, видеозаписей и музыкальных композиций, затем используется для анализа комментариев, относящихся, скажем, к потребительской электронике: персональным компьютерам и планшетам. Сделаем обобщение: можно представить себе, что существует базовая функция, которая выдает эмоциональную окраску – положительную, нейтральную или отрицательную. Конечно, в данной ситуации словарный состав и стиль могут зависеть от предметной области – домена, что затрудняет обобщение с одного домена на другой. Простое обучение без учителя показало прекрасные результаты при анализе эмоциональной окраски посредством

адаптации домена, а также во многих других задачах, связанных с обработкой естественного языка [37, 38, 39].

В данной работе были использованы модели, натренированные на различные задачи (например маскирование текста), которые обладают высокой обобщающей способностью, а также модели, наученные на текстах другого домена (принадлежащие смежному домену математических текстов или широкой выборке обычных текстов).

Модели, использующие перенос знаний и адаптации домена:

a) GloVe - натренирован на корпусе составленном из Википедии, новостных лентах и Твиттере [40];

b) Word2Vec – натренирован на корпусе составленном из Google Новостей (около 100 млрд слов) [41];

c) USE - натренирован на корпусе SNLI [42];

d) BERT – натренирован на BooksCorpus [43] и английской Википедии;

e) MathBERT – натренирован на связанных с математикой корпусах документов, которые включают в себя учебные программы по математике от дошкольного до старшего школьного; учебники, написанные для старшеклассников и студентов колледжей; программы курсов по математике из публичных открытых онлайн-курсов; рефераты по математике; также аннотации математических LaTeX статей [44].

4.4.2. Стратифицированная перекрестная проверка

Разделение набора данных на фиксированные обучающий и тестовый становится проблематичным, если в тестовом наборе оказывается слишком мало данных. Малый тестовый набор – причина статистической недостоверности в оценке средней ошибки тестирования, из-за которой трудно утверждать, что алгоритм «А» работает лучше алгоритма «В» на конкретной задаче [28].

В случае, когда набор данных насчитывает сотни тысяч и более примеров, то особой проблемы не возникает. Но когда набор данных мал, можно прибегнуть к альтернативным методам, которые позволяют использовать все примеры для оценивания средней ошибки тестирования ценой увеличения объема вычислений. В качестве примера такого метода используется перекрестная проверка.

Процедура перекрестной проверки заключается в следующем. Фиксируется некоторое множество разбиений исходной выборки на две части: обучающую и контрольную. Для каждого разбиения выполняется настройка алгоритма по обучающей подвыборке и вычисляется частота его ошибок на контрольной подвыборке. Оценка

скользящего контроля определяется как среднее по всем разбиениям частота ошибок на контрольной выборке [45].

Стратифицированная перекрестная проверка обычно полезна, когда у нас имеется набор несбалансированных данных и когда размер данных невелик. Основная идея заключается в сохранении дисбаланса классов, когда он представляет или содержит некоторую информацию о природе объекта, который мы пытаемся классифицировать.

Когда данные достаточно велики, по-прежнему существует возможность использовать обычную перекрестную проверку, поскольку это сохраняет соотношение классов, но такая ситуация становится менее вероятным событием с меньшим количеством обучающих примеров. Например, предположим, что у нас есть проблема бинарной классификации, где на один класс приходится 90% обучающих примеров, но у нас есть миллион обучающих примеров. В данной ситуации, обычная перекрестная проверка, наиболее вероятно, сохранит соотношение между классами. Рассмотрим ситуацию если бы у нас была всего тысяча обучающих примеров для одной и той же задачи. В этом случае более вероятно, что разные разбиения на подмножества будут иметь разные соотношения классов, особенно с большими значениями количества разбиений, и могут даже привести к разбиению, содержащему обучающие примеры только из одного класса.

В данной работе использовалась стратифицированная перекрестная проверка реализованная в scikit-learn [46].

4.4.3. F1-мера

Существуют задачи, в которых необходимо построить модель бинарной классификации, обнаруживающей редкие события. Примером такой задачи служит тест для диагностики редкого заболевания. Построим пример, где заболевание встречается у одного человека из очень большой выборки людей, например из миллиарда. Для этой задачи определения класса объекта легко получить 99.99% вероятности правильно распознать объект, просто заставив классификатор всегда сообщать об отсутствии заболевания у пациента. Данный пример наглядно демонстрирует проблемы несбалансированности классов и оценки классификатора.

Введем сначала понятие матрицы ошибок. Это способ разбить объекты на четыре категории в зависимости от комбинации истинного ответа и ответа алгоритма.

Таблица 3 – Матрица ошибок

	$y = 1$	$y = 0$
$f(x) = 1$	TP (True Positive)	FP (False Positive)
$f(x) = 0$	FN (False negative)	TN (True Negative)

где x – входные данные модели (признаки объекта);

$f(x)$ – метка класса, предсказанная моделью;

y – метка класса предсказываемого объекта.

В таблице 3 отображена матрица ошибок, через элементы которой можно, например, выразить долю правильных ответов – accuracy (5.1).

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (5.1)$$

Очевидно, что метрика accuracy (5.1) не подходит для оценки качества такой системы, которая определяет вероятность правильной классификации объекта, т.к. не является информативной для задачи определения пациента с редким заболеванием, и, как показано выше в примере, мы легко можем получить на ней высокую точность заставив модель в любой ситуации, независимо от входных данных, предсказывать доминирующий класс.

Решить проблему можно, если измерять не accuracy, а полноту (recall) (5.2) и точность (precision) (5.3).

$$\text{Recall} = \frac{TP}{TP+FN} \quad (5.2)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (5.3)$$

Точностью называется доля правильных ответов модели, а полнотой – доля обнаруженных истинных событий. Классификатор, утверждающий, что нет ни одного больного, имеет идеальную точность, но нулевую полноту. Классификатор, утверждающий, что больны все, достигает идеальной полноты, но его точность равна процентной доле людей, страдающих заболеванием (0,0001 в случае, когда заболеванием страдает один человек на миллион). Отметим отдельно, что точность и полнота не зависят от соотношения размеров классов.

Существует несколько способов получить один критерий качества на основе точности и полноты. В данной работе рассмотрена F-мера [47]. F-мера (5.4) является средним гармоническим значением точности и полноты.

$$\text{F1} = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (5.4)$$

Среднее гармоническое обладает важным свойством — оно близко к нулю, если хотя бы один из аргументов близок к нулю. В случае, когда объектов положительного

класса на порядки меньше, чем объектов отрицательного класса, данная метрика будет корректно отображать качество работы алгоритма, что является информативным показателем для представленной в работе задачи классификации определений.

5. Результаты экспериментов

В данном разделе представлены результаты обучения классификаторов определений для математических текстов формата LaTeX различных векторных представлений. Все результаты были получены с помощью метода перекрестной проверки.

Таблица 4 – Результаты работы классификаторов

Векторное представление	Модель классификатора	F-мера	Accuracy
BOW	Градиентный бустинг	0.44259	0.97605
TF-IDF	Градиентный бустинг	0.36139	0.97314
Word2Vec	Градиентный бустинг	0.20350	0.97419
GloVe	Градиентный бустинг	0.26854	0.97644
USE	Градиентный бустинг	0.29525	0.95790
BERT	Градиентный бустинг	0.35244	0.97118
MathBERT	Полносвязная искусственная нейронная сеть	0,22522	0.93059
MathBERT	Градиентный бустинг	0.48798	0.97495

В Таблице 4 представлены результаты работы классификаторов на векторных представлениях, полученных посредством различных моделей.

Можно увидеть, что наибольшее качество получила модель MathBERT, натренированная на корпусах математического текста. Это обосновывается тем, что механизм внимания данной модели наиболее приспособлен к различению отношений между математическими объектами, за счет домена-источника (корпуса математических документов).

Из двух подходов к классификации, более плохое качество на модели MathBERT получил классификатор из полносвязной нейронной сети (данный подход был применен, сообразно классическому варианту использования [21]), по сравнению с моделью градиентного бустинга, что обуславливается малым набором данных, где, для данной задачи, градиентный бустинг показывает более высокие результаты.

Также можно увидеть, что достаточно высокое качество (2-ое по ранжированию F-меры) показала модель BOW, что обосновывается общей разреженностью текста и неупорядоченным подходом к его анализу (см. п. 4.1, 4.3).

Заключение

В рамках магистерской диссертации было проведено исследование методов машинного обучения и обработки естественного языка в задаче бинарной классификации предложений на «определение» и «не определение» из математических LaTeX статей.

Был проведен анализ текущего состояния предметной области, который показал общую неисследованность темы применения алгоритмов машинного обучения для «сырых» LaTeX документов. Был произведен разведочный анализ, построены визуализации векторных представлений, размечены математические LaTeX статьи, после чего построены классификаторы, основанные на различных векторных представлениях, с использованием градиентного бустинга и применением искусственных нейронных сетей.

Разведочный анализ языковых структур математических документов показал равномерную распределенность терминов и формальных конструкций языка LaTeX в тексте статей.

Анализ визуализации векторных представлений показал, что векторизованные «определения» имеют сильно отличающуюся природу (точки, выделенные на графиках расположены далеко друг от друга), что не позволяет добиться высокого качества работы классификатора на данной задаче.

Проведенные вычислительные эксперименты показали, что среди построенных классификаторов, основанных на различных векторных представлениях, моделями с наименьшим качеством работы являются: Word2Vec, GloVe, USE. Данный факт свидетельствует о низком качестве адаптации домена классических текстов на домен математических текстов.

Наиболее высоким качеством работы классификации обладает полученное векторное представление модели MathBERT с классификатором на основании градиентного бустинга.

В качестве перспективных направлений для будущего исследования стоят задачи:

a) Дообучения MathBERT на математических статьях, заменяя формулы токеном, то есть дообучение на задаче маскирования текста, для получения более устойчивых и соответствующих задаче векторных представлений с последующей проверкой на NTCIR-12;

b) Увеличения объема существующего набора данных, для получения более статистически достоверных результатов;

с) Сбора набора данных для обучения автокодировщиков с целью очистки его от служебных слов LaTeX, которые не вносят контекстуальной нагрузки;

Также, с помощью вышеуказанных пунктов, планируется развитие задачи классификации «определений» на предложениях в задачу извлечения именованных сущностей: определяемых понятий и символов, описывающих их.

Библиографический список:

1. Ganesalingam M. The language of mathematics //The language of mathematics. – Springer, Berlin, Heidelberg, 2013. – С. 272.
2. Официальный сайт LaTeX [Электронный ресурс]. – Режим доступа: https://www.latex-project.org/, свободный. – Загл. с экрана.
3. Ohri A., Schmah T. Machine translation of mathematical text //IEEE Access. – 2021. – Т. 9. – С. 38078-38086.
4. Berlioz L. ArGoT: A Glossary of Terms extracted from the arXiv //arXiv preprint arXiv:2109.02801. – 2021.
5. Berlioz L. WIP: Creating a Database of Definitions From Large Mathematical Corpora.
6. Zhang D. et al. The gap of semantic parsing: A survey on automatic math word problem solvers //IEEE transactions on pattern analysis and machine intelligence. – 2019. – Т. 42. – №. 9. – С. 2287-2305.
7. Kriaučiukas V., Razinkovas L., Žamoitinaite L. Parsing LATEX for LATEX //BachoTEX 2015 proceedings. – С. 31-36.
8. Chuvilin K. Machine learning approach to automated correction of LaTeX documents //2016 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT). – IEEE, 2016. – С. 33-40.
9. Официальный сайт LaTeXML [Электронный ресурс]. – Режим доступа: https://math.nist.gov/~BMiller/LaTeXML/, свободный. – Загл. с экрана.
10. Agrawal R. et al. Fast algorithms for mining association rules //Proc. 20th int. conf. very large data bases, VLDB. – 1994. – Т. 1215. – С. 487-499.
11. Park S. H. et al. Apriori-based text mining method for the advancement of the transportation management plan in expressway work zones //The Journal of Supercomputing. – 2018. – Т. 74. – №. 3. – С. 1283-1298.
12. Jones K. S. A statistical interpretation of term specificity and its application in retrieval //Journal of Documentation: MCB University: MCB University Press. – 2004. – Т. 60. – №. 5. – С. 493– 502.
13. Qaiser S., Ali R. Text mining: use of TF-IDF to examine the relevance of words to documents //International Journal of Computer Applications. – 2018. – Т. 181. – №. 1. – С. 25-29.
14. Harris Z. S. Distributional structure //Word. – 1954. – Т. 10. – №. 2-3. – С. 146-162.

15. Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.

16. Mikolov T. et al. Distributed representations of words and phrases and their compositionality //Advances in neural information processing systems. – 2013. – Т. 26.

17. Аггарвал Ч. Нейронные сети и глубокое обучение: учебный курс //Санкт-Петербург: ООО «Диалектика. – 2020. – С. 752.

18. Gutmann M., Hyvärinen A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models //Proceedings of the thirteenth international conference on artificial intelligence and statistics. – JMLR Workshop and Conference Proceedings, 2010. – С. 297-304.

19. Morin F., Bengio Y. Hierarchical probabilistic neural network language model //International workshop on artificial intelligence and statistics. – PMLR, 2005. – С. 246-252.

20. Pennington J., Socher R., Manning C. D. Glove: Global vectors for word representation //Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). – 2014. – С. 1532-1543.

21. Vaswani A. et al. Attention is all you need //Advances in neural information processing systems. – 2017. – Т. 30.

22. Iyyer M. et al. Deep unordered composition rivals syntactic methods for text classification //Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers). – 2015. – С. 1681-1691.

23. Cer D. et al. Universal sentence encoder for English //Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations. – 2018. – С. 169-174.

24. Devlin J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding //arXiv preprint arXiv:1810.04805. – 2018.

25. Peng S. et al. Mathbert: A pre-trained model for mathematical formula understanding //arXiv preprint arXiv:2105.00377. – 2021.

26. Официальный сайт NTCIR-12 [Электронный ресурс]. – Режим доступа: https://ntcir-math.nii.ac.jp/task-overview/, свободный. – Загл. с экрана.

27. Van der Maaten L., Hinton G. Visualizing data using t-SNE //Journal of machine learning research. – 2008. – Т. 9. – №. 11.

28. Бенджио И., Гудфеллоу Я., Курвилль А. Глубокое обучение //Москва: ДМК-Пресс. – 2018. – С. 401

29. Friedman J. H. Stochastic gradient boosting //Computational statistics & data analysis. – 2002. – Т. 38. – №. 4. – С. 367-378.

30. Chen T., Guestrin C. Xgboost: A scalable tree boosting system //Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. – 2016. – С. 785-794.

31. Официальный сайт Kaggle [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/, свободный. – Загл. с экрана.

32. Официальный сайт Brat [Электронный ресурс]. – Режим доступа: https://brat.nlplab.org/, свободный. – Загл. с экрана.

33. Официальный сайт Arxiv API [Электронный ресурс]. – Режим доступа: https://brat.nlplab.org/, свободный. – Загл. с экрана.

34. Официальный сайт Inception [Электронный ресурс]. – Режим доступа: https://inception-project.github.io/, свободный. – Загл. с экрана.

35. Официальный сайт PdfTeX [Электронный ресурс]. – Режим доступа: https://tug.org/applications/pdftex/, свободный. – Загл. с экрана.

36. Akiba T. et al. Optuna: A next-generation hyperparameter optimization framework //Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. – 2019. – С. 2623-2631.

37. Glorot X., Bordes A., Bengio Y. Domain adaptation for large-scale sentiment classification: A deep learning approach //ICML. – 2011.

38. Do C. B., Ng A. Y. Transfer learning for text classification //Advances in neural information processing systems. – 2005. – Т. 18.

39. Raina R., Ng A. Y., Koller D. Constructing informative priors using transfer learning //Proceedings of the 23rd international conference on Machine learning. – 2006. – С. 713-720.

40. Официальный сайт GloVe [Электронный ресурс]. – Режим доступа: https://nlp.stanford.edu/projects/glove/, свободный. – Загл. с экрана.

41. Официальный сайт SNLI [Электронный ресурс]. – Режим доступа: https://nlp.stanford.edu/projects/snli/, свободный. – Загл. с экрана.

42. Официальный сайт Word2Vec со ссылками на данные для обучения [Электронный ресурс]. – Режим доступа: https://code.google.com/archive/p/word2vec/, свободный. – Загл. с экрана.

43. Zhu Y. et al. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books //Proceedings of the IEEE international conference on computer vision. – 2015. – С. 19-27.

44. Github репозиторий MathBERT со ссылками на данные для обучения [Электронный ресурс]. – Режим доступа: https://github.com/tbs17/MathBERT, свободный. – Загл. с экрана.

45. Воронцов К. В. Комбинаторный подход к оценке качества обучаемых алгоритмов //Математические вопросы кибернетики. – 2004. – Т. 13. – С. 5-36.

46. Официальный сайт scikit-learn [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/index.html, свободный. – Загл. с экрана.

47. Sasaki Y. et al. The truth of the F-measure //Teach tutor mater. – 2007. – Т. 1. – №. 5. – С. 1-5.