Федеральное государственное автономное образовательное
учреждение высшего образования
«Московский физико-технический институт
(государственный университет)»
Физтех-школа прикладной математики и информатики
Кафедра дискретной математики

Направление подготовки: 01.04.02 Прикладная математика и информатика

ПРИМЕНЕНИЕ СОВРЕМЕННЫХ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ИЕРАРХИЧЕСКОЙ СУММАРИЗАЦИИ НАУЧНЫХ ТЕКСТОВ

(магистерская диссертация)

Студент:
Костин Александр Александрович
Научный руководитель:
Воронцов Константин Вячеславович
д-р физ.-мат. наук, профессор

Москва
2026

Содержание

Обозначения и сокращения

- *БЯМ* — большая языковая модель (large language model).
- *Chain-of-Thought, CoT* — цепочка рассуждений БЯМ.
- *ИИ* — искусственный интеллект.
- *NLP* — обработка естественного языка (natural language processing).
- *ROUGE* — Recall-Oriented Understudy for Gisting Evaluation.
- *State-of-the-Art, SOTA* — передовой уровень, последние достижения в определенной области знаний.
- *Chunk, Чанк* — дословно «кусок» или порция текста, формируемая для отдельной обработки автоматизированными средствами.

Введение

Актуальность темы. Еще каких-то 20–30 лет назад одним из основных вызовов для человека был дефицит знаний: чтобы написать курсовую или диплом, нужно было идти в библиотеку, штудировать бумажные подборки периодики и монографии. Сегодня ситуация кардинально изменилась, - объемы информации в современном мире растут в геометрической прогрессии. По некоторым данным, за последние два года было создано более 90% всей информации, существующей на планете. Можно сказать, что мы живем в эпоху информационного цунами.

Многие люди сталкиваются с тем, что их мозг физически не приспособлен обрабатывать гигантские объемы информации ежедневно, что приводит к колоссальному разрыву между скоростью поступления информации и нашей физической способностью ее усвоить. Чтение текста средней сложности требует времени: на вдумчивое прочтение среднего объема научной статьи уходит от 15 минут до 3-4 часов, а на книгу – несколько дней. При этом задача человека, в особенности – исследователя, остается прежней – ему нужно понимать суть анализируемого материала и принимать правильные решения, но делать это быстрее.

Именно здесь на помощь приходит суммаризация (или создание саммари, сводки) – это процесс выделения сути, ключевых мыслей и выводов из массива данных, главное преимущество которой – экономия времени. Для анализа научных статей это особенно важно, создание сводок помогает бороться с "информационным шумом", а умение быстро извлекать суть становится в современном мире таким же важным навыком, как умение читать и писать.

Следует отметить, что каждая научная статья обязательно имеет аннотацию ("abstract"), очень краткую сводку того, о чем написана статья. В то же время, аннотация не позволяет исследователю погрузиться в материал, узнать те или иные подробности по ключевым идеям и находкам авторов, сравнить их с другими концепциями и подходами. Для решения таких задач разработаны различные методологические подходы, одним из самых эффективных представляется иерархическая суммаризация, или создание иерархических сводок.

Исследования в области информатики показали, что люди не читают документ линейно при создании краткого содержания или сводки, они используют структуру: сначала просматривают заголовки верхнего уровня, затем переходят к разделам, которые кажутся наиболее важны-

ми, и углубляются до тех пор, пока не найдут достаточное количество ключевых предложений. Авторы статьи “Hierarchical Summarization of Large Documents” называют это “Outline Extraction Behavior” [1]. Существующие на тот момент (2008 год) автоматические модели рассматривали документ как простую последовательность предложений и строили сводку линейно, выбирая большую часть предложений из самых “тяжелых” (важных) разделов, что приводило к информационной избыточности и потере разнообразия тем (теряются важные, но менее объемные подтемы из других разделов).

Авторы предложили построить процесс построения иерархической сводки рекурсивно сверху вниз: система рассчитывает “важность” (т.н. фрактальное значение) каждого узла (раздела) и распределяет лимит предложений (квоту) пропорционально этой важности. Этот процесс повторяется до тех пор, пока на самом нижнем уровне не будут отобраны конкретные предложения на основе статистических признаков (тематических, позиционных и др.). Стоит также отметить, что авторы применили подходы, очень близкие к тому, как работают БЯМ на основе технологии трансформера (механизм внимания и пр.). При этом, метод был использован только для одно-документальной суммаризации.

Подход иерархической суммаризации для большой коллекции документов был предложен Кристенсен Ж. в работе 2014 года [2]. Новая парадигма, по мнению авторов, похожа на то, как человек, интересующийся сложной темой, узнает о ней от эксперта: сначала эксперт дает общее представление, а затем более подробную информацию о различных аспектах. Иерархическая суммаризация, фактически строя дерево из смыслов, извлеченных из корпуса текстов, по принципу “от общего к частному”, обладает следующими характеристиками:

- Обобщение иерархически организовано в соответствии с одним или несколькими организационными принципами, такими как время, местоположение, объекты или события;
- Каждая ветка иерархической сводки, не являющаяся листом дерева, связана с набором дочерних ветвей, каждая из которых содержит подробную информацию об элементе (например, предложении) в родительской ветви;
- Пользователь может перемещаться по иерархической сводке, нажимая на элемент родительской ветви, чтобы просмотреть соответствующие дочерние ветви.

Иерархическая сводка имеет два важных преимущества: во-первых, объем информации, представленной в начале, минимально необходимый и увеличивается только по мере того, как пользователь обращается к нижестоящим ветвям дерева, что не перегружает пользователя; во-вторых, каждый пользователь сам определяет зону своего интереса, поэтому ему нужно изучить только этот раздел данных, не просматривая и не анализируя всю сводку целиком, а ссылки между родительскими и дочерними элементами позволяют ему перемещаться по данным, углубляясь в интересующие темы.

Популярность набирает и построение интеллект-карт – один из способов оформления иерархических сводок для удобного графического восприятия. Одним из родоначальников такого подхода является Тони Бьюзен [3], который так описывает процесс в своей книге: “Собственно техника построения интеллект-карт - это один из наиболее эффективных инструментов визуализации. Она разработана как развитие мнемонических техник, таких, как метод локусов (“дворец памяти”) известных еще со времен античности. У хорошей интеллект-карты три основные составляющие: центральный образ, передающий тему (предмет) изучения, основные ветви (ключевые темы), отходящие от центрального изображения и единственное ключевое слово или изображение на каждой ветви."

В статье “Mapping Scientific Frontiers: The Quest for Knowledge Visualization” авторы также указывают, что визуализация знания играет критическую роль в истории развития науки и техники, обеспечивая отображение процесса перехода от абстрактного мышления к конкретному. [4]

В контексте настоящей работы и применительно к сводкам научных статей следует не согласиться только с подходом автора о “единственном ключевом слове” для каждой вершины (или ветви) интеллект-карты. Для погружения в материал научной работы ни слова, ни даже словосочетания недостаточно. Представляется, что наиболее удобным было бы превращение иерархической сводки (или интеллект-карты, как ее графического представления) в стройное единое повествование от общего к частному, в которое исследователь погружается по мере “открытия” все новых уровней интеллект-карты.

В этой связи Константин Вячеславович Воронцов отмечает, что “Переход от текста к нелинейному чтению меняет привычные способы потребления информации, способствует массовому развитию навыков выделять во всем главное, достигая взаимопонимания и консенсуса при

коммуникации.” [5]

Итак, какова же цель использования иерархических сводок и интеллект-карт по научным публикациям? Ответ на этот вопрос не представляется однозначным. В ходе исследования на семинарах «Мастерская знаний» в Институте искусственного интеллекта МГУ[1] этот вопрос обсуждался многократно и у разных участников всегда были свои взгляды на проблему и цели составления интеллект-карт. Вместе с тем, без фиксации ответа на вопрос представляется принципиально невозможным решить задачу иерархической суммаризации, поскольку сама иерархия будет строиться исходя из разных принципов.

По мнению автора, можно выделить основную цель для использования интеллект-карт в научном домене – понимание материала (можно еще это назвать структурированным запоминанием). Причем, просто сокращение текста до 5-10% (или даже до 150 слов аннотации) от его первоначального объема понять статью не помогает, а вот интеллект-карты, как раз, здесь подходят очень хорошо, что подтверждает ряд исследователей.

Необходимо отметить, что статьи часто пишутся отчасти уже как линейный пересказ иерархической суммаризации, обычно трехуровневой:

- аннотация — верхний уровень — авторы стремятся отразить основные достижения и привлечь внимание аудитории к своей работе;
- введение — второй уровень — раскрывается предыстория проблемы, мотивация, повторяются основные мысли из аннотации, иногда чуть более детализированно;
- основной текст – третий уровень, максимальная детализация – авторы вынуждены в третий раз повторять ряд основных тезисов, что представляется особенным неудобством для читающего, поскольку приходится вспоминать и искать то место статьи, где была представлена “вот та” очень удачная формулировка.

Иерархическая суммаризация устраняет необходимость таких повторов.

Во многих источниках суммаризация подразделяется на три направления: экстрактивная, абстрактивная и гибридный подход. Экстрактивная суммаризация – извлечение из исходного текста наиболее информативных предложений. Абстрактивная – генерация текста сводки с учетом

[1] «Мастерская знаний» - научный проект Института искусственного интеллекта МГУ под руководством д.ф.-м.н. Воронцова К.В.

морфологии, синтаксиса, семантики, благодаря чему формируется логически и по смыслу связный текст, что дает этому способу второе название – “глубинный” [6]. По мнению автора, именно абстрактивный подход к построению иерархических сводок наиболее полно отвечает на запрос о понимании (структурированном запоминании) научного текста.

Авторы статьи SummEval: Re-evaluating Summarization Evaluation [7] предлагают следующие критерии успешной сводки:

- Связность — общее качество текста резюме. Как указывает Dang H.T. [8], «резюме должно быть хорошо структурировано и организовано. Резюме должно представлять собой не просто набор связанных между собой сведений, а последовательное изложение информации по теме»;
- Согласованность — это фактическое соответствие между резюме и источником, на основе которого оно составлено, и содержать только те утверждения, которые вытекают из исходного документа;
- Плавность (беглость) — это качество отдельных предложений в резюме (отсутствие грамматических и пунктуационных ошибок, стилистических небрежностей);
- Релевантность — резюме должно включать только важную информацию из исходного документа.

При этом можно сказать, что текстовым форматом для интеллект-карты является иерархическая сводка. На семинарах «Мастерской знаний» были выработаны и принципы, которым должна соответствовать иерархическая сводка, составленная для решения вышеуказанной задачи:

- Значимость – отобрать главное (концепции, идеи), отранжировать по важности, что дает возможность построить иерархическую структуру сводки, и, соответственно, метод передачи знаний;
- Однородность – подтемы образуют повествование (нарратив), связанны единым сюжетом статьи (или более обобщенно – корпуса статей), что позволяет читать иерархическую сводку, последовательно спускаясь от уровня к уровню;
- Полнота – подтемы охватывают все аспекты текста, по которому делается сводка, в которой не остается лакун и пробелов;

- Точность – среди подтем невозможно выделить лишнюю, критерий, связанный с «полнотой» и обеспечивающий неизбыточность (non-redunancy) сводки.

На указанных семинарах также выделялись требования «эргономики», относящиеся не к содержанию, а оформлению интеллект-карт по иерархическим сводкам, и упомянуть о которых стоит ввиду обозначенных выше в настоящем исследовании целей использования интеллект-карт для обеспечения понимания материала: наглядность, подкрепляемая изображениями рядом с текстом; лаконичность как требование формулировать темы максимально кратко.

Практическая польза от применения интеллект-карт в образовательном процессе подтверждается рядом авторов. Так, в статье «Mind map learning technique: an educational interactive approach» [9] отмечается, что проведенный эксперимент с использованием интеллект-карт при обучении студентов сестринскому делу (медицина) показал, что группа студентов, применявшая новые методики, уверенно лидировала по среднему баллу, и, кроме того, получила навыки критического мышления при разрешении клинических случаев.

Авторы книги Towards an AI-Augmented Textbook [10] из LearnLM Team, Google в рамках подхода “Learn Your Way” предлагают множество техник обогащения учебного контента, интеллект-карты являются одной из них. Это полностью автоматизированный, персонализированный и интерактивный инструмент, глубоко интегрированный в учебный процесс. Он перекладывает тяжелую работу по анализу и синтезу структуры материала с человека на ИИ, позволяя ученику сосредоточиться на понимании и запоминании, используя карту как навигатор и опорный конспект. Этот подход является значительным шагом вперед по сравнению с ручными и статичными методами создания интеллект-карт.

Другими исследователями [11] проведено сравнение обычного чтения научных статей с подчеркиванием наиболее важных частей текста, использованием выносок и комментариев с составлением интеллект-карт, которое, по результатам проведенных экспериментов, увеличило продуктивность специалистов в среднем на 50%.

Подробный обзор использования интеллект-карт в высшем образовании представили Митра А. и др. [12], также отметив высокие результаты обучающихся, применявших карты в учебе.

Стоит отметить, что в приведенных публикациях авторы проводили эксперименты с составлением самими обучающимися интеллект-карт,

что давало дополнительные возможности для усвоения и понимания материала. При этом, с учетом современных инструментов в виде больших языковых моделей, даже составленные «внешним агентом» в виде алгоритма интеллект-карты будут способствовать эффективному пониманию области научных интересов у исследователей, и, безусловно, экономить значительную часть времени, затрачиваемого на прочтение, анализ и понимание публикаций.

Цель и задачи работы. Целью настоящего исследования является составление интеллект-карт (точнее иерархических сводок как текстового представления интеллект-карты) для понимания научных публикаций с помощью современных автоматизированных средств.

Задачи работы:

- Исследовать подходы к суммаризации научных статей и их развитие во времени.
- Установить, могут ли БЯМ быть использованы для создания иерархических сводок как самодостаточный инструмент.
- Предложить новый алгоритм создания иерархических сводок, приближенных по качественным характеристикам к эталонным.
- Провести сравнение сгенерированных предложенным алгоритмом сводок с имеющимися автоматизированными аналогами.

Научная новизна. Научная новизна работы заключается в разработке гибридного метода иерархической суммаризации научных текстов, объединяющего графовое представление смысловой структуры документа и возможности больших языковых моделей. В отличие от классических методов суммаризации, ориентированных на линейный отбор наиболее значимых предложений, в работе предлагается строить сводку как иерархическое дерево тем, подтем и деталей. Новым является использование трехуровневой схемы: локальные семантические миниграфы, объединенный рабочий граф и итоговое иерархическое дерево. Предложенный подход позволяет явно фиксировать смысловые связи между фрагментами научного текста, а не только оценивать их статистическую или позиционную значимость.

Особенностью работы является то, что большая языковая модель используется не как единственный генератор итоговой сводки, а как кон-

тролируемый модуль извлечения, уточнения и редактирования поверх детерминированного графового каркаса. Это снижает зависимость результата от произвольной генерации модели и повышает интерпретируемость алгоритма. В работе предложен метод комбинирования двух механизмов построения иерархии: выделения тематических сообществ в графе и использования типизированных направленных отношений между вершинами.

Важным отличием предлагаемого метода является обязательное сопоставление каждой вершины итоговой иерархии с опорными предложениями исходного текста. Такой механизм позволяет проверять обоснованность включения тем и подтем в сводку и уменьшает риск появления неподтвержденных обобщений. Дополнительной новизной является ориентация метода именно на научные тексты, где важны не только краткость и связность, но и сохранение логики исследования, терминов, аргументов, результатов и ограничений. В отличие от GraphRAG-подобных подходов, граф в данной работе используется не как индекс для ответа на внешний запрос, а как центральный объект построения самой иерархической сводки. Тем самым работа предлагает интерпретируемый и проверяемый алгоритм автоматического построения текстового представления интеллект-карты научной публикации.

Результаты, выносимые на защиту. На защиту выносятся следующие основные результаты работы:

1. Предложен гибридный метод иерархической суммаризации научных текстов, обеспечивающий построение сводки в виде дерева тем, подтем и деталей и отличающийся совместным использованием графового представления документа и больших языковых моделей.

2. Предложена трехуровневая схема представления текста «локальные семантические миниграфы — объединенный рабочий граф — итоговое иерархическое дерево», обеспечивающая переход от фрагментарного анализа отдельных частей статьи к целостной структуре научного содержания и отличающаяся явным сохранением промежуточных графовых артефактов.

3. Предложен способ построения иерархии по рабочему графу, обеспечивающий формирование структуры от общих тем к частным деталям и отличающийся сочетанием двух механизмов: выделения

тематических сообществ и анализа направленных типизированных связей между вершинами.

4. Предложен механизм сопоставления вершин дерева итоговой иерархической сводки с опорными предложениями исходного текста, обеспечивающий проверяемость включенных в сводку тем и отличающийся снижением риска появления неподтвержденных или слабо обоснованных обобщений.

5. Предложена схема итеративного уточнения иерархической сводки с помощью БЯМ, обеспечивающая улучшение формулировок, удаление избыточных фрагментов и повышение связности дерева и отличающаяся тем, что БЯМ используется не как единственный генератор результата, а как контролируемый редактор поверх графового каркаса.

6. Реализован и экспериментально проверен прототип алгоритма `minigraph`, обеспечивающий автоматическое построение иерархических сводок научных публикаций и отличающийся интерпретируемостью промежуточных этапов, возможностью выбора режима построения локального графа и сопоставимостью результатов с экспертной разметкой.

1 Анализ предметной области

Стоит отметить, что практически все работы по много-документальной суммаризации не конкретизируют ее цель, ограничиваясь общими формулировками о необходимости обработки большого объема текстовой информации [13]. В то же время, применяемые подходы довольно сильно зависят от цели. Например, для понимания научных текстов, особенно для целого корпуса документов, иерархия представления сущностей и их связей, представляется необходимой.

Целью суммаризации является создание краткой и информативной сводки Sum из набора документов D. D обозначает кластер тематически связанных документов $\{d_i \mid i \in [1, N]\}$, где N — количество документов. Каждый документ d_i состоит из M_{d_i} предложений $s_{i,j} \mid j \in [1, M_{d_i}]$. $s_{i,j}$ относится к j-му предложению в i-м документе. Эталонное резюме называется «золотым стандартом». Авторы отмечают, что в настоящее время большинство сводок «золотого стандарта» составляется экспертами [13].

В исследовании Ли и др. [14] проведено сравнение экспертных сводок и сводок, подготовленных различными моделями, специально предобученными под задачу суммаризации. Алгоритмы просто привели набор фактов из исходного текста, в то время как эксперты вывели общий смысл, сконструировав уровень абстракций не используя цитат из документов.

Большинство работ по суммаризации посвящены составлению сводок по новостным статьям, описанию событий, знаменитостей – метод, как правило, представляет собой сведение простых сущностей воедино во избежание повторов и одновременно быстрого погружения в контекст. На изученность методов обобщения научных статей в меньшей степени, чем новостей, указывает и ряд исследователей [15]. Суммаризация научных статей очевидно сложнее суммаризации новостей, а цитаты в последующих работах показывают, какие именно аспекты статьи оказались наиболее востребованными [15].

Для научных статей составление сводок приносит, по мнению автора, гораздо больше пользы – изучение огромных объемов информации, выделение главного для исследования и взгляд с разных сторон на изучаемый предмет, проблему. При этом, научные статьи намного семантически и структурно сложнее: объем текстов больше, многоаспектнность, описание различных подходов, сравнение с другими методами и экспериментальный анализ. Задача составления сводки в данном случае может

быть сформулирована: как сократить корпус научных текстов так, чтобы не потерять главное для исследователя, сохранив при этом нужный уровень детализации?

Задача сокращения научных текстов для быстрого изучения предметной области появилась с ростом объема знаний, и, как следствие, с увеличением числа статей, прочтение которых требуется исследователю для погружения в материал. Использовать автоматизированные средства для решения данной задачи одним из первых предложил Luhn H.P. в 1958 году в статье The Automatic Creation of Literature Abstracts [16] – им был разработан статистический алгоритм для автоматического составления аннотаций к научным статьям. Программа выполнялась на IBM 704, специально сконструированной для работы с данными. Вычисляя частоту встречаемости слов в тексте, алгоритм вычислял фактор значимости каждого предложения, причем наиболее значимым считалось предложение с наибольшим числом частотных слов, расположенных максимально близко друг к другу внутри предложения. Далее наиболее значимые предложения помещались в аннотацию в том порядке, в каком они расположены в тексте статьи.

При этом, автор уже тогда отмечал, что алгоритм не может быть универсальным ввиду специфики материалов в различных областях, и предлагал применять метод для подготовки аннотаций для статей в области технологий и естественных наук.

Одной из первых попыток построения интеллект-карт из текста является работа Абдин и др. [17], использующая классические NLP-методы (вместо статистических в предыдущей цитируемой работе): разбиение слов на морфемы, синтаксический анализатор на основе WordNet, семантическая обработка текста для выбора правильного значения слова, дискурс-анализ для привязки местоимений к существительным, извлечение глаголов как признаков события, существительных и прилагательных, связанных с событием.

Рассмотренный подход был доработан авторами статьи [18], введены как многоуровневый алгоритм обобщения текста, так и выделение из него т.н. «фреймов» для сущностей и действий, связанных определенными типами отношений.

В названной выше работе Кристенсен Ж. [2] одним из первых предложен подход к построению иерархических сводок. Ван и др. [19] предложили новую модель для суммаризации научных статей, интегрируя подход к тексту как линейной последовательности и существующую иерар-

хию разделов статьи, а также локальную связность предложений внутри одной темы (секции, подзаголовка).

Лапата и др. [20] предложили подход к суммаризация одного большого документа путем разбиения его на небольшие смысловые части (chuncks), их суммаризации и далее построения общей сводки с помощью алгоритма объединения/замещения контекста.

С появлением больших языковых моделей (БЯМ) ряд практических задач, в том числе задачи суммаризации текстов стали решаться с помощью них. В статье Summarization is (Almost) Dead [21]: авторы сравнили аннотации, сделанные людьми, специально обученными моделями (BART, T5, Pegasus) и большими языковыми моделями (GPT-3, GPT-3.5, GPT-4) и пришли к выводу, что БЯМ показывают существенно лучшие результаты с точки зрения стилистической корректности и точности изложения фактов (отсутствия "галлюцинаций"), в том числе по оценке самих людей. Эксперимент проводился в том числе и в мультидокументальном формате.

Действительно, БЯМ сделали серьезный рывок во многих задачах, но оценка в работе проводилась, в основном, на текстах новостей и программном коде, что представляется не применимым к научным публикациям, особенно с учетом их объема и задачи построения иерархической аннотации для обеспечения понимания предметной области.

При этом, ряд авторов отмечают, что современные БЯМ хорошо справляются с генерацией текста, написанием кода, но они не умеют явно моделировать сложные отношения между концепциями из разных статей. При простой конкатенации сводок и запросе на генерацию общего обзора БЯМ либо теряет информацию (особенно при длинном контексте), либо генерирует поверхностный текст, перечисляя работы без логических связей [22].

Это и является открытой проблемой, не разрешенной до конца на текущий момент. Целью настоящего исследования является поиск методов улучшения задачи генерации иерархических сводок с помощью БЯМ, максимального приближения качества генерации к уровню сводок, составленных человеком.

Современные нейросетевые абстрактивные модели (обычно на базе Seq2Seq и Transformer) хорошо работают для одного документа, но при переходе к нескольким документам (MDS) сталкиваются с трудностями выявления кросс-документных связей. Модели обрабатывают документы независимо или как простую конкатенацию, теряя связи между пред-

ложениями из разных источников (повторы, противоречия, дополнения). Сгенерировать связное и краткое изложение из множества источников сложно без понимания структуры связей [23].

Чжэн Я. и др. [24] провели глубокое исследование суммаризации и указывают на то, что традиционные методы были жестко привязаны к одной парадигме: либо экстрактивной, либо абстрактивной, а выбор архитектуры модели и данных предопределял способ генерации. В то же время БЯМ, благодаря своим фундаментальным свойствам (обучение на огромных корпусах данных, способность к обучению в контексте), могут переключаться между этими парадигмами или смешивать их в рамках одной задачи без переобучения, что, по мнению авторов, является новой революцией в суммаризации текстов.

Работа [24] наиболее полно классифицирует методы суммаризации с помощью БЯМ, выделяя следующие:

- Промпт-инжиниринг, формализуемый как $S = LLM_{\Phi}(G(p)+x)$, где $G(p)$ – улучшенный промпт. В этой группе методов следует особо выделить пошаговое рассуждение (Chain-of-Thought, CoT), например «найди важные сущности, даты, события, а потом объедини», и мультиагентные системы, когда несколько БЯМ взаимодействуют между собой в разных ролях, например: генератор, инструктор, редактор;

- Генерация, дополненная извлечением информации (Retrieval-Augmented Generation, RAG), в том числе графовые методы;

- Тонкая настройка (Fine-tuning): обновление малой части параметров (например, только слои внимания), или предобучение адаптеров на большом корпусе текстов и дообучение на целевом;

- Дистилляция знаний (Knowledge Distillation), или перенос способностей от большой БЯМ (учитель) к компактной модели (студент).

Вместе с тем, соверменные БЯМ достаточно сильно подвержены “галлюцинациям” или придумыванию информации, не существовавшей в исходном тексте, что может привести к серьезным искажениям сводки. Нарайян Ш. и др. [25] указывают, что традиционные методы суммаризации использовали промежуточное представление (план) для управления генерацией сводки, но в нейросетевых подходах этот этап отсутствует, и предлагают восстановить планирование, используя цепочки сущностей –

упорядоченный список (имена, даты, числа), которые должны появиться в итоговой сводке. Фактически, авторы задействуют механизм пошагового рассуждения (Chain-of-Thought, CoT), на первом этапе пошагово выделяя сущности, а на втором – добавляя в промпт список сущностей и сводки-кандидаты. Представляется такой подход достаточно интересным, но применимым прежде всего к новостям, на которых он прежде всего тестировался авторами.

Обширное практическое исследование использования цепочек рассуждения при составлении запроса к БЯМ провели Вей Дж. и др. [26], установив, что способность БЯМ выдавать правильные ответы в задачах, требующих рассуждения или просто комплексных, значительно увеличивается. Авторы показывают модели не только пример из вопроса и ответа по аналогичной задаче, но и промежуточные шаги рассуждения на естественном языке, которые ведут к ответу. При тестировании модель генерирует аналогичную цепочку мыслей, а затем финальный ответ, что позволяет раскладывать сложную задачу на последовательность простых шагов и приходить к верному ответу. Важно заметить, что цепочка рассуждений хорошо работает на моделях с большим числом параметров (более 100 млрд.), а для моделей с числом параметров менее 10 млрд. результат даже ухудшается.

Левиафан Я. и др. [27] обнаружили, что простое повторение всего пользовательского запроса (промпта) дважды улучшает точность ответов БЯМ, не увеличивая длину генерируемого текста и не повышая задержку (latency). Эффект объясняется причинно-следственной архитектурой БЯМ (causal LM), когда каждый токен может “видеть” только предыдущие токены. Повторение позволяет каждому токену первого экземпляра промпта “увидеть” второй экземпляр (и наоборот) благодаря механизму самовнимания, что дает модели более полный контекст. Авторы отмечают серьезный прирост производительности, но только для не-рассуждающих (Non-Reasoning) моделей, в последних же похожий механизм запускается автоматически в ходе цепочки рассуждений.

Чен и др. [28] предложили подход CoTHSSum для суммаризации длинных документов, который объединяет иерархическую сегментацию входного текста и цепочку рассуждений БЯМ (Chain-of-Thought) на каждом таком сегменте. Это позволяет БЯМ сначала логически разобрать содержание по частям, а затем структурированно собрать итоговую сводку, вместо обработки сразу всего документа за один проход. Такой подход снижает галлюцинации, улучшает полноту охвата и связность сводки.

Подход к построению иерархических сводок с помощью БЯМ был смоделирован и апробирован при участии автора[2] в рамках работы проекта «Мастерская знаний», показав, что при генерации иерархической сводки с помощью ряда последовательных запросов к модели, получается достичь результатов, близких к сводкам, составленным людьми. Тем не менее, этот метод имеет ряд ограничений: большой расход токенов, и, как следствие, высокая стоимость подготовки одной сводки, и невозможность использовать уже полученные в рамках анализа других статей знания (формирование каждой сводки “с нуля”).

Таким образом, для решения задачи иерархической суммаризации недостаточно простого использования БЯМ, требуется дополнительная работа с семантическими, смысловыми конструкциями текста, начиная от их выявления, установления связей между ними и иерархии «от общего к частному».

[2] `https://gitlab.mlsa-iai.ru/mindmaps/LLM-single-doc-summary`

2 Графовые структуры при создании иерархических сводок

Для суммаризации корпуса документов ряд исследователей поднимают следующие важные вопросы [13]:

- Как извлекать междокументные и внутридокументные связи?
- Как извлекать значимую информацию в большем массиве текста, зачастую содержащем противоречивые, дублирующие и дополняющие друг друга данные?
- Как наилучшим образом объединять различные представления, полученные с помощью моделей глубокого обучения и внешних знаний?

При построении иерархических сводок научных статей важно учитывать следующие аспекты:

- Необходимо единообразно структурировать суммаризации разных документов, выделяя общие аспекты: проблема, предлагаемый метод, теория, результаты и т.д.
- Для научных статей важно учитывать временную ось - отличать современные методы (State-of-the-Art, SOTA) от устаревших подходов.

Понимая важность иерархии для подготовки сводки можно представить два варианта подходов к ее построению. Во-первых, это формальная структура, которая в большинстве научных публикаций изначально присутствует: мотивация, проблематика, недостатки существующих подходов, идея исследования, модели и методы, проверяемые гипотезы, методика эксперимента, полученные результаты, интерпретации. Следующим уровнем при таком подходе является детализация, формальные подразделы разделов. Фактически, линейное изложение статьи раскладывается в структуру, которая уже присутствует в исходном тексте. При этом, целью такой структуры является представление авторами публикации своего результата научной общественности и доказательство разработки авторами новой единицы знаний. Именно эта цель и делает такой вариант иерархии ограниченным, поскольку пользователь сводки решает задачу структурированного понимания материала.

Второй вариант иерархии – это начать с главного, сформулировать самые важные проблемы и единицы знаний, которые содержит анализируемая статья. Причем, не обязательно ограничиваться только SOTA подходами, и не обязательно только результатами, полученными авторами статьи. Критериями в данном случае являются важность для применения результатов, продолжения и расширения исследований, в целом для науки и прогресса цивилизации. Новизна и обоснование авторского вклада в науку отходят на второй план. Детализация иерархии объясняет основные понятия, концепции, идеи, которые могут остаться непонятными из лаконичной формулировки идеи первого уровня, при этом ставится цель максимально быстро добраться до ответа на вопрос «что надо сделать, чтобы это использовать идею из статьи», причем очень важно – какие проблемы, накладывающие ограничения на использование, остались открытыми. По мнению автора, второй подход к построению иерархии при суммаризации научной статьи наиболее отвечает поставленной задаче.

Интерес в данном контексте представляет работа Таухманн и др. [29], которая предлагает методологию создания корпуса для иерархической суммаризации, где информация организована в виде деревьев. Корень дерева содержит общие сведения по теме, а ветви разбиваются на конкретные аспекты (facets), детализируясь до конкретных фактов, мнений, аргументов. Такая структура позволяет получать как обобщенные сводки (верхние уровни), так и детальные сводки по каждому аспекту исследуемой темы. Для создания корпуса предлагается двухэтапный процесс: сначала краудсорсинг для сбора релевантных информационных единиц (nuggets), затем экспертная аннотация для их иерархической организации. Сам корпус текста, подготовленный авторами, составлен по достаточно узким темам из медицины и психологии, но этот подход создания «золотого стандарта» или эталона для оценки автоматических иерархических суммаризаторов, был использован автором и рабочей группой «Мастерской знаний» при Институте искусственного интеллекта МГУ. Так, были подготовлены иерархические сводки по 20 научным статьям, охватывающим, в основном, тему иерархической суммаризации и связанные научные вопросы.

Очевидно, что иерархия для рассматриваемой задачи суммаризации может быть на абстрактном уровне отлично положено на графовые конструкции, особенно с учетом того, что иерархическая сводка, как и интеллект-карта, представляет собой дерево, что является одним из ви-

дов графов.

Основными типами графов, применяемых для задач анализа текста, являются «графы дискурса» (discourse graph) и «графы знаний» (knowledge graph).

Одно из первых формальных определений графа дискурса было сделано Кристенсен Ж. и др. [30] – это ориентированный граф предложений, где ребра задают порядок, необходимый для связности сводки. В последующей работе [2] развивались варианты этого подхода, которые будут рассмотрены далее. Чан Дж. [31] описывает дискурсивный граф как модель, где «представление знаний сосредоточено на единицах утверждений и их связях». Майазоно и Акамацу [32] указали на роль графа дискурса в структурировании научных аргументов, когда каждая вершина графа является вопросом, утверждением или доказательством, связанными ребрами.

Термин «граф знаний» популяризирован компанией Google в 2012 году при запуске собственного Knowledge Graph для обогащения результатов поиска [33]. Существует несколько вариантов определения графа знаний:

- Граф данных, предназначенный для накопления и передачи знаний о реальном мире, где узлы — сущности, ребра — отношения [34];
- Крупные сети сущностей, их семантических типов, свойств и отношений [35];
- RDF-граф, состоящий из троек (субъект, предикат, объект) [36];
- Архитектура, включающая онтологию и механизм вывода новых знаний [37].

В чем же принципиальные отличия и в чем похожи два типа графов? Граф дискурса моделирует связи между фрагментами текста, представляя текст как их набор – от слова до предложений, реплик или документов, связанных отношениями, например, пояснения, противопоставления, причины, уточнения. Главная цель данной структуры – это зафиксировать логико-риторическую организацию текста, поэтому иерархия в нем часто присутствует изначально или выводится сравнительно естественно.

Граф знаний моделирует связи между понятиями, сущностями и фактами, которые являются вершинами графа, ребра же выступают как

источник информации о взаимодействии понятий, иногда с указанием силы связи, полярности или опоры на конкретные фрагменты источника. Граф знаний делает явной предметную структуру содержания текста, особенно когда нужно объединять сведения из нескольких документов.

Рассмотренные различия показывают, что для задачи суммаризации научных текстов больше подходит граф знаний.

Одной из первых попыток применить графовую структуру для суммаризации стала работа Кристенсен Ж. Авторы работы представляют свою систему G-Flow[3] как независимую от области применения (новости, стенограммы встреч или научные тексты), что представляется как слабая сторона подхода применительно к теме настоящего исследования [30].

Ясунага М. и др. [38] в 2017 году предложили более совершенный подход, применив персонализированный дискурсивный граф (Personalized Discourse Graph, PDG), добавив к весам ребер “персонализацию т.е. вес каждого ребра умножили на оценку предложения (персонализированный коэффициент), полученную с помощью линейной регрессии на основе 8 макро-признаков (позиция в документе, наличие имен собственных, длина и т.д.). Это делает веса более разнообразными и информативными для сети. Кроме того, была применена графовая сверточная сеть (Graph Convolutional Network, GCN) для обогащения представлений (эмбеддингов) предложений на основе структуры графа, и, как следствие, учет при выборе наиболее важных предложений связей с остальной частью корпуса текстов. В то же время, авторы опять же работают с предложениями вместо смысловых единиц текста (концепций, ключевых терминов и др.), что представляется не самым удачным подходом.

Переход к работе с параграфами как единице, более отвечающей задаче поиска смысловых единиц обрабатываемого текста, предложен в работе Ли и др. [23]. Такой подход все равно нельзя признать идеальным, поскольку деление на структурные единицы текста (параграфы) не всегда отражает концептуальный смысл научного текста и не позволяет представить его в виде иерархической сводки. Вместе с тем, впервые в данной работе показано, как «встроить граф в архитектуру (Transformer)», добавив его в универсальный механизм внимания (Self-Attention). Кроме того, работа Ли успешно решает проблему масштабирования на длинные последовательности, добавляя предобученные модели (BERT/RoBERTa) в качестве мощного средства извлечения призна-

[3] `https://knowitall.cs.washington.edu/gflow/README.html`

ков для узлов графа (параграфов).

Ху М. и др. [39] предложили не просто построение графа отношений из предложений исходного документа, но и уточнение такого графа с помощью алгоритмов обучения с подкреплением (Reinforcement learning) на основе выделения людьми наиболее важных предложений. При этом, непосредственное построение интеллект-карты происходит путем выбора вершины графа для корня дерева и двух дочерних вершин (по суммам весов ребер, исходящих из вершины), с повторением данного процесса уже для дочерних вершин.

Ван П. и др. [40] предложили подход, использующий граф знаний и модель архитектуры трансформер с дополнительным элементом, названным KGtext generator, который превращает граф знаний в текстовое представление сущностей и их отношений. Результат работы генератора используется при формировании итоговой сводки по корпусу научных статей, причем авторы заметили, что лучший результат достигается при использовании при работе KGtext generator всех трех компонент графа: сущности, их типы, отношения между сущностями, что является еще одним подтверждением полезности информации из графа знаний для подготовки итоговых сводок.

Чжэн Ч. и др. [41] строят интеллект-карты, используя промежуточное представление оригинального текста в виде графа кореференции, что позволяет на основе упоминания одних и тех же сущностей (имен, названий и др.) построить иерархию (первое, самое важное упоминание и далее последующие упоминания сущности в тексте, раскрывающие важные детали). Представляется, что такой подход является значительно упрощенным и неприменимым к научным текстам, прежде всего как не позволяющий учесть связи между различными сущностями (научными концепциями, терминами, методами).

Определенный интерес представляет работа Лу Дж. и др. [42], предлагающая построение концептуальных карт из набора документов без дополнительного обучения. Метод также основан на использовании графа, в качестве вершин которого выделяются сущности (осмысленные фразы из документов, например, «deep learning», «moon landing») с помощью стандартных инструментов библиотеки Stanford CoreNLP. Ребра графа как связи между сущностями проводятся, если в скользящем окне (несколько предложений, параметр настраиваемый) две вершины встречаются, вес ребра при этом определяется на основе частотности встречаемости. Далее авторы применяют двухслойную графовую сверточную

сеть из работы Кипф, Уиллинг и др. [43] для построения представлений вершин графа с учетом контекста. Далее Graph Translator (GPT), являющийся ядром и инновацией метода, преобразует начальный граф в итоговый, генерируя последовательность узлов и соответствующие векторы связей в ходе итеративного процесса, а итоговый граф (последовательность узлов и матрица смежности) подается в Graph Isomorphism Network (GIN) для получения векторного представления всего графа.

Вместе с тем, предложенный метод генерирует не сводки по текстам, а только концепт-карты, которые не удовлетворяют цели суммаризации, рассматриваемой в настоящем исследовании, т.е. понимания научных текстов пользователем.

Научные публикации, как правило, имеют стройную логическую структуру (введение, сопутствующие работы, метод, эксперименты, обсуждение, заключение и будущая работа), которую можно было бы использовать для генерации графа, но такой подход содержит ряд минусов: во-первых, внутри каждого из разделов может содержаться (и обычно содержится) более одной смысловой единицы, во-вторых, установление отношений в графе знаний между разделами из разных статей будет невозможно.

Ряд исследователей [33] отмечает, что будущее за глубокой интеграцией графов знаний с БЯМ для повышения точности, обоснованности и объяснимости результатов, а автоматическое построение графов знаний с использованием самообучения и методов глубокого обучения приведет к улучшению масштабируемости и снижению ручного труда при извлечении сущностей и отношений между ними.

Интересный подход предложили Чжэн Ч. и др. [22], впервые используя БЯМ в качестве гибкого экстрактора сущностей и отношений, причем эта задача выполняется динамически, с учетом контекста всей темы, что позволяет адаптировать граф под конкретную задачу и не требует отдельной обученной SciIE-системы, как в предыдущих работах. Основу подхода составляют Knowledge Minigraph Construction Agent (KMCA) и Multiple Path Summarization Agent (MPSA).

KMCA строит небольшой, тематически сфокусированный граф (minigraph), а не пытается охватить все возможные связи. Это решает проблему избыточности и шума от массы нерелевантной информации, характерную для общих графов знаний. Проблема длинного контекста для БЯМ решается разбиением корпуса статей на отдельные части (chunks) и последовательным обновлением графа, что принципиаль-

но отличается от рассмотренного подхода KGSum, где все документы обрабатывались сразу, но с ограничением по длине. Итеративность позволяет сохранить информацию из большого числа источников без потери качества.

В отличие от KGSum, где генерация идет по единственному пути (хотя и с учетом графа), MPSA генерирует несколько альтернативных сводок, соответствующих разным логическим цепочкам в графе, выбирая лучшую с помощью метрики ROUGE, что позволяет получить более структурированный и связный текст.

Таким образом, представленный подход представляет собой синтез идей графового представления и развитых возможностей БЯМ по обработке текста, интегрируя извлечение информации и генерацию сводки в единый процесс на основе БЯМ, делая систему более гибкой и масштабируемой. Использование мини-графов, итеративного построения и роутера решает проблемы, с которыми сталкивались рассмотренные выше методы: шумность данных в больших графах, ограничение длины контекста языковых моделей и неоднозначность структуры изложения итоговой сводки.

Некоторые исследователи [44] предлагают подход к построению графа знаний, основанный на онтологии, т.е. набору из 219 свойств, которому должны соответствовать все ребра в графе. Например, для ребра «cskg:textbook-dataset, uses, cskg:spatial-prediction», вершина «cskg:textbook-dataset» - это экземпляр класса «cskg-ont:Material», а вершина «cskg:spatial-prediction» - это экземпляр класса «cskg-ont:Task», а значит, свойство «uses» неприменимо - его доменом не является «cskg-ont:Material».

Представляется, что столь подробная и сложная онтология как раз является слабой стороной метода, ведь большое число свойств могут быть избыточными для некоторых задач, пользователю нужно дорабатывать онтологию для применения к нужной области знаний (онтология и сам граф CS-KG 2.0 построен авторами на основе публикаций в сфере компьютерных наук).

К минусам предложенного метода также необходимо отнести ограниченность источников для построения графа только заголовками и аннотациями статей, полные тексты не анализируются из-за технических ограничений, что может приводить к потере детальной информации (например, из раздела экспериментов) и не соответствовать главной задаче построения иерархических сводок, обозначенной в настоящем исследовании – понимании материала из корпуса научных текстов.

Первичное построение графа знаний без использования заданной онтологии является достаточно сложной задачей. Романова А. [45] демонстрирует, что даже простые графовые представления в сочетании с графовыми нейронными сетями могут давать интересные результаты для анализа текстов. Вместе с тем, если в подходах SciERC/SciER/CS-KG используются сложные обученные модели (DyGIE++, SciBERT), то у Романовой А. - простой метод на основе коллокаций и стоп-слов, что дает менее точные и семантически бедные сущности, обладая только преимуществом быстрого построения (возможно, для целей прототипирования) графов без разметки и использования даже минимальной схемы.

Интересным представляется метод SciIE (Scientific Information Extractor) [46], представляющий собой единую модель, одновременно (multi-task) решающую три задачи для создания графа знаний:

- Распознавание сущностей (Named Entity Recognition) — выделение ключевых терминов (методы, задачи, материалы);
- Извлечение отношений (Relation Extraction) — определение связей между сущностями (например, «метод A используется для задачи B»);
- Разрешение кореференции (Coreference Resolution) — связывание упоминаний одной и той же сущности в разных местах текста (включая местоимения и общие термины).

Ангелов Д. и др. [47] предлагают решение для построения тематической модели коллекции документов, в которой каждый документ сегментирован на монотематические кластеры, каждая тема описана списком из N релевантных фраз, а само число тем оптимизируется параметрами модели. Представляется, что такой подход для выделения вершин в графе знаний не подходит, поскольку сущности (идеи, концепции) могут значительно детализироваться внутри найденных тем.

3 Реализация гибридного подхода с использованием графов и БЯМ

3.1 Постановка задачи и общая схема метода

Рассмотрим корпус научных статей

$$\mathcal{D} = \{d_i\}_{i=1}^{M},$$

где каждый документ d_i представим как упорядоченную последовательность абзацев

$$d_i = \{p_{i,j}\}_{j=1}^{N_i},$$

где каждому абзацу сопоставлены текст, название секции и, при наличии, номер секции. После сегментации абзацев на предложения получаем множество предложений

$$\mathcal{S} = \bigcup_{i=1}^{M} \mathcal{S}_i, \qquad \mathcal{S}_i = \{s_{i,t}\}_{t=1}^{T_i}.$$

Требуется построить иерархическую сводку в виде дерева

$$\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, r),$$

где корень r соответствует названию статьи или общему тематическому заголовку корпуса текстов, а каждая вершина $v \in V_{\mathcal{T}}$ задается тройкой

$$v = \big(y(v), \mathcal{E}(v), \mathrm{Ch}(v)\big),$$

где $y(v)$ — текстовая формулировка вершины, $\mathcal{E}(v) \subseteq \mathcal{S}$ — множество опорных предложений (*evidence*), а $\mathrm{Ch}(v)$ — множество дочерних вершин.

Искомая иерархическая сводка должна удовлетворять следующим требованиям:

- иерархичность: верхние уровни должны описывать более общие темы, а нижние — их детали, компоненты, свойства, результаты или ограничения;
- однородность: дерево, будучи читаемым обходом от корня, раскрывается как повествование (нарратив), что позволяет читать иерархическую сводку, последовательно спускаясь от уровня к уровню;

- полнота: в дереве представлены все аспекты текста, по которому делается сводка, в которой не остается лакун и пробелов, все вершины сопоставлены опорным предложениям из текста;
- не избыточность: одинаковые или почти одинаковые смысловые конструкции не должны дублироваться по разным ветвям.

В целом, задачу можно записать как поиск дерева

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \Big(\lambda_{\text{hier}} \,\text{Hier}(\mathcal{T}) + \lambda_{\text{nar}} \,\text{Narr}(\mathcal{T}) + \lambda_{\text{full}} \,\text{Full}(\mathcal{T}) - \lambda_{\text{red}} \,\text{Red}(\mathcal{T}) \Big),$$

где:

- Hier(T) — мера иерархичности дерева;
- Narr(T) — мера нарративной однородности дерева, то есть того, насколько дерево читается как последовательное связное раскрытие темы при обходе от корня к листьям;
- Full(T) — мера полноты, учитывающая как покрытие смысловых аспектов исходного текста, так и наличие опорных предложений для вершин дерева;
- Red(T) — мера избыточности, отражающая степень смыслового дублирования между вершинами и ветвями.

Дополнительно для корпуса текстов может строиться граф цитирования, однако в текущей реализации он не используется для построения иерархической сводки.

На вход алгоритму подаются статьи в формате Semantic scholar (S2ORC) JSON, который содержит не только полный текст статей, разбитый на отдельные абзацы с идентификатором раздела статьи, но и полный граф цитирования, а также другие полезные метаданные[4].

Предлагаемый алгоритм состоит из семи основных этапов:

1. фильтрация секций и сегментация текста на предложения;
2. разбиение текста на чанки контролируемого размера;
3. извлечение из каждого чанка локального семантического минграфа с помощью LLM;
4. слияние локальных миниграфов в единый мультиорграф;

[4]Вспомогательная работа и алгоритмы описаны в приложении А

5. вычисление значимости вершин и построение рабочего графа G^*;
6. построение детерминированного корневого иерархического дерева (черновика иерархической сводки);
7. итеративное уточнение структуры сводки с помощью LLM с обязательной проверкой опоры на предложения из текста.

Рассмотрим все шаги алгоритма более подробно.

3.2 Предварительная обработка и разделение текста на чанки

Нормализация и фильтрация секций. Для работы с текстом научной статьи необходимо исключить те разделы (секции), которые с большой долей вероятности не несут информации о концепциях и идеях авторов, и, одновременно, могут генерить шум для составления сводки. Обычно это разделы приложений, обзора литературы, список литературы[5].

Пусть $\sigma(p)$ — название секции абзаца p. Введем оператор нормализации

$$\mathrm{NormSec}(\sigma) = \mathrm{collapse\_spaces}\big(\mathrm{lower}(\mathrm{strip\_leading\_section\_number}(\sigma))\big).$$

Нормализация необходима, чтобы разные записи одного и того же заголовка привести к единому виду и включает приведение к нижнему регистру, удаление лишних пробелов, а также удаление числового префикса раздела.

Тогда для набора стоп-секций

$$\Sigma_{\mathrm{drop}} = \{\texttt{acknowledgements}, \texttt{references}, \texttt{bibliography}, \texttt{appendix}, \texttt{relatedworks}\}$$

абзац удаляется из дальнейшей обработки, если выполнено хотя бы одно из условий:

$$\exists \tau \in \Sigma_{\mathrm{drop}} : \ \tau \subseteq \mathrm{NormSec}(\sigma(p)),$$

или

$$\mathrm{NormSec}(\sigma(p)) \text{ соответствует шаблону заголовка таблицы,}$$

или

$$\mathrm{TopLevelSec}(p) \in \Pi_{\mathrm{drop}},$$

[5]Поскольку большая доля статей публикуются на английском языке, названия секций, применяемые шаблоны и другие лексические конструкции используются для английского языка

где $\mathrm{TopLevelSec}(p)$ — название раздела верхнего уровня по отношению к секции, к которой принадлежит абзац p, Π_{drop} — множество верхнеуровневых числовых префиксов тех секций, которые были помечены как исключаемые. Такой порядок означает, что если исключается, например, секция «2 Related Work», то при включенном режиме наследования исключаются и подсекции вида «2.1», «2.2» и т.д.

Сегментация на предложения. Для каждого абзаца p выполняется разбиение на предложения

$$p \mapsto \{s_1, \ldots, s_K\}.$$

Система поддерживает два режима, выбор которых производится в конфигурационном файле:

1. разбиение на основе регулярных выражений по шаблону конца предложения

   `(?<=[.!?])\s+(?=[A-ZA-ЯЁ0-9])`,

2. сегментацию средствами научной модели spaCy/SciSpaCy.

SciSpaCy — это библиотека и набор моделей для процессинга научных текстов, использование которой снижает число ошибочных разрезов в местах, где в статьях встречаются сокращения, специальные обозначения, ссылочные конструкции и нестандартная пунктуация. Регулярные выражения могут пропускать такие случаи, что приводит к ошибкам сегментации и дальнейшего сопоставления идентификаторов предложений, опорных предложений и чанков, вызывая каскад ошибок. Проведенные тесты разных режимов показали, что SciSpaCy обеспечивает меньшую долю ошибок.

Каждому предложению присваивается уникальный идентификатор

$$\mathrm{id}(s) = \texttt{paper\_id:s}\langle\text{6-значный номер}\rangle,$$

Выявление предложений, перегруженных ссылками. Предложения, перегруженные ссылочными конструкциями, как правило, не отражают идей автора и могут создавать шумные данные, делающие сводку менее связной, поэтому такие предложения либо удаляются, либо получают уменьшающий их значимость признак, либо могут быть сохранены без изменений.

Для предложения s вычисляется плотность цитирования

$$\rho_{\text{ref}}(s) = \frac{\sum\limits_{m \in \mathcal{M}(s)} |m|}{|s|},$$

где $\mathcal{M}(s)$ — множество совпадений с шаблонами:

1. числовые ссылки вида [1], [1, 2, 5];
2. выражения вида *et al.*;
3. ссылки формата *Автор (Год)* или *(Автор, Год)*.

Предложение считается перегруженным ссылками, если

$$\rho_{\text{ref}}(s) \geq 0.15.$$

После сегментации предложения группируются по исходным абзацам:

$$B_{i,j} = \{s_{i,t} \colon \text{ paragraph\_index}(s_{i,t}) = j\}.$$

Пусть

$$W(X) = \text{число слов в текстовом фрагменте } X.$$

Тогда для абзаца

$$W(B_{i,j}) = \sum_{s \in B_{i,j}} W(s).$$

Агрегация последовательных абзацев. Чанк c строится как последовательность соседних абзацев одной и той же секции, с учетом следующих ограничений:

$$W_{\min}^{\text{target}} = 150, \qquad W_{\max} = 350, \qquad W_{\min} = 40, \qquad W_{\text{overlap}} = 30,$$

где $W_{\min}^{\text{target}}$ — целевой нижний порог объема абзаца в словах, $W_{\min}$ — жесткий нижний порог текстового фрагмента, оставшегося при «разрезании» абзацев. Второй параметр задает минимальный осмысленный фрагмент текста.

Пусть $B_1, B_2, \ldots$ — последовательные абзацы одной статьи и одной секции. Тогда новый чанк формируется жадным алгоритмом до достижения максимального размера чанка по словам:

$$c = B_t \cup B_{t+1} \cup \cdots \cup B_{t+\ell},$$

пока выполняются условия

$$W(c) < W_{\min}^{\text{target}} \quad \text{и} \quad W(c \cup B_{t+\ell+1}) \le W_{\max}.$$

Абзацы из разных секций не смешиваются. Для секции *Abstract* используется отдельный поток чанков: аннотация не смешивается с основным текстом статьи.

Разрезание слишком больших абзацев. Если абзац слишком велик,

$$W(B_{i,j}) > W_{\max},$$

он разрезается не по символам, а по предложениям. Для контроля тематической связности используется мера Жаккара по содержательным токенам:

$$J(s_a, s_b) = \frac{|T_{\text{cont}}(s_a) \cap T_{\text{cont}}(s_b)|}{|T_{\text{cont}}(s_a) \cup T_{\text{cont}}(s_b)|},$$

где $T_{\text{cont}}(s)$ — множество токенов предложения без стоп-слов и без слишком коротких слов.

Разрез предпочтительно ставится в точке, где одновременно выполнены условия

$$W(\text{буфер}) \ge W_{\min} \quad \text{и} \quad J(s_{t-1}, s_t) < \tau_{\text{coh}}, \qquad \tau_{\text{coh}} = 0.1.$$

Если тематический сдвиг не обнаружен, то разрез ставится по ограничению $W_{\max}$.

Перекрытие чанков. Для улучшения локальной связности (когда одна и та же мысль автора статьи могла быть разделена на разные чанки по принципам деления по предложениям) осуществляется перекрытие соседних чанков. Пусть c_{k-1} — предыдущий чанк. Тогда в начало нового чанка переносится хвост из последних предложений предыдущего чанка:

$$\text{Tail}(c_{k-1}) = \{s_t, \ldots, s_m\},$$

где предложения выбираются с конца до тех пор, пока суммарное число слов не достигнет W_{overlap}.

Итоговый чанк имеет вид

$$c_k = \text{Tail}(c_{k-1}) \cup \text{Core}(c_k).$$

3.3 Локальный семантический минограф

Определение. Для каждого чанка c_k строится локальный типизированный ориентированный граф

$$G_k = (V_k, E_k),$$

где

$$V_k = \{v_{k,1}, \ldots, v_{k,n_k}\}$$

— вершины, соответствующие сущностям (понятиям, концепциям или кратким утверждениям) в тексте статьи, а E_k — типизированные ребра с опорой на предложения.

Построение локальных графов не опосредовано какой-то онтологией, поскольку такой подход не обладает достаточными обобщающими свойствами на любую область знаний. При этом, для соблюдения упорядоченности отношений между сущностями, множество допустимых типов отношений ограничено:

$$\mathcal{R} = \{\mathsf{used\_for}, \mathsf{feature\_of}, \mathsf{hyponym\_of}, \mathsf{part\_of}, \mathsf{compare}, \mathsf{conjunction}, \mathsf{evaluate\_for}\}.$$

Каждое ребро представляется как

$$e = (u, r, v, \mathcal{E}_e),$$

где $u, v \in V_k$, $r \in \mathcal{R}$, $\varnothing \neq \mathcal{E}_e \subseteq \mathcal{S}(c_k)$, а $\mathcal{S}(c_k)$ — опорные предложения данного чанка, «доказывающие» связь между сущностями-вершинами локального графа.

Извлечение локального графа с помощью БЯМ. Извлечение локального графа может производиться с помощью БЯМ или эвристически. При извлечении с помощью БЯМ, в последнюю, помимо системного и пользовательского запросов передаются следующие данные:

1. текст чанка;
2. список идентификаторов $\mathrm{id}(s)$ опорных предложений чанка;
3. заданное множество типов отношений $\mathcal{R}$.

БЯМ согласно требованиям из системного промпта, возвращает JSON-объект с вершинами и ребрами, удовлетворяющий следующим ограничениям:

$$|V_k| \leq 50, \qquad |E_k| \leq 200,$$

при этом каждая метка вершины должна быть короткой фразой длины от 2 до 8 слов, а каждое ребро должно ссылаться на 1—3 идентификатора опорных предложений чанка.

После получения ответа от БЯМ выполняется проверка на наличие ребер с недопустимыми типами, а также наличие вершин без инцидентных ребер. Такие ребра и вершины удаляются:

$$E'_k = \{e \in E_k \colon \operatorname{type}(e) \in \mathcal{R}\},$$

$$V'_k = \{v \in V_k \colon \exists e \in E'_k,\ v = \operatorname{src}(e) \ \vee \ v = \operatorname{dst}(e)\}.$$

Эвристический режим извлечения локального графа. Пусть

$$\operatorname{tf}_k(t)$$

— частота токена t в чанке c_k, причем используются только токены длины не менее 4. Тогда вершины локального графа задаются как восемь наиболее частотных токенов:

$$V_k = \operatorname{Top}_8\big(\operatorname{tf}_k\big).$$

Ребра строятся цепочкой, а тип связи выбирается по типу раздела статьи, к которому относится чанк, если же секция не распознана, используется тип связи conjunction.

Этот подход является базовым. В то же время, проведенные эксперименты (Приложение В. Сравнение иерархических сводок, созданных различными алгоритмами и системами с «золотым стандартом») показали, что подход к извлечению локального графа с помощью БЯМ работает качественнее, но незначительно, при этом, по структурным метрикам уступая эвристике.

3.4 Слияние локальных графов

Все локальные графы объединяются в единый ориентированный мультиграф

$$G = (V, E),$$

где допускаются множественные ребра между одной и той же парой вершин.

Канонизация меток вершин. Перед слиянием метка вершины проходит канонизацию (приведение к каноническому виду):

$$\kappa_i(v) = A_i(\mathrm{Canon}(v)),$$

где Canon включает:

1. удаление ведущих и хвостовых пробелов;
2. удаление двойных и множественных пробелов;
3. приведение к нижнему регистру;
4. приведение аббревиатур по словарю из библиотеки SciSpaCy: A_i : short form $\mapsto$ long form.

Объединение ребер. Если режим объединения параллельных ребер графа выключен в конфигурационном файле, то каждое ребро между двумя вершинами локального графа добавляется в G как отдельная дуга:

$$e = (u, r, v, \mathcal{E}_e, \mathrm{chunk\_id}, \mathrm{paper\_id}).$$

Если объединение включено, то все ребра с одинаковым ключом сливаются в одно ребро, а его атрибуты объединяются.

3.5 Расчет значимости вершин и построение рабочего графа

Сбор опорных предложений для вершин и ребер объединенного графа. Для каждой вершины $v \in V$ собирается множество опорных предложений со всех инцидентных ребер:

$$\mathcal{E}(v) = \bigcup_{e \ni v} \mathcal{E}_e.$$

Для каждого ребра e запоминается число уникальных опорных предложений:

$$c_e = |\mathcal{E}_e|.$$

Построение неориентированной проекции графа. В мультиграфе G между одной и той же парой вершин могут существовать несколько дуг, возможно разных типов и из разных частей текста (чанков). Для вычисления грубых структурных характеристик вершины такой граф является

излишне подробным: ориентация и параллельные дуги здесь затрудняют решение задачи определения, насколько вершина вообще связана с другими вершинами и насколько она играет роль «моста» между частями графа. В то же время, для целей расчета центральности в алгоритме важна не конкретная направленность дуги, а сам факт наличия связи между двумя понятиями, концептами. Поэтому из ориентированного мультиграфа G строится простой неориентированный граф

$$U = (V_U, E_U), \qquad V_U = V.$$

Для любых двух различных вершин $u, v \in V$ вводится величина

$$m(u,v) = \bigl|\{\, e \in E : \ e \text{ соединяет } u \text{ и } v, \ u \neq v \,\}\bigr|,$$

равная числу дуг между u и v в исходном мультиграфе без учета направления. Тогда

$$\{u,v\} \in E_U \iff m(u,v) > 0, \qquad w_U(u,v) = m(u,v).$$

Иначе говоря, каждая пара связанных вершин представляется одним неориентированным ребром, а его вес равен числу параллельных дуг между этими вершинами в G. Изолированные вершины добавляются в U отдельно, чтобы сохранить полный набор вершин при вычислении метрик важности вершин графа для иерархической сводки.

На графе U далее вычисляются две структурные характеристики вершины:

$$d(v) = \mathrm{DegreeCentrality}_U(v), \qquad b(v) = \mathrm{BetweennessCentrality}_U(v).$$

Первая величина отражает число различных соседей вершины и тем самым характеризует ее локальную связанность, тогда как вторая измеряет «посредническую роль» вершины, то есть ее участие в кратчайших путях между другими вершинами графа. Обе метрики вычисляются по топологии графа U; накопленные веса $w_U(u,v)$ при вычислении центральностей не используются.

Поскольку далее структурные и неструктурные признаки объединяются в единую оценку важности вершины графа, каждый скалярный признак x_v приводится к шкале $[0,1]$ с помощью min–max-нормировки:

$$\mathrm{mm}(x_v) = \begin{cases} 1, & \max_u x_u - \min_u x_u < \varepsilon, \\ \dfrac{x_v - \min_u x_u}{\max_u x_u - \min_u x_u}, & \text{иначе.} \end{cases}$$

Если признак вырожден и принимает одинаковое значение для всех вершин, то после нормировки всем вершинам присваивается значение 1.

Секционный приоритет и взвешенная сумма важности опорных предложений. После объединения локальных миниграфов каждой вершине v сопоставляется множество опорных предложений

$$\mathcal{E}(v) \subseteq \mathcal{S},$$

из которых данная вершина (как понятие, концепция из анализируемой статьи) была извлечена, либо с которыми она оказалась связана при объединении локальных графов. Однако сами по себе опорные предложения неоднородны по своей информативной ценности: предложения из аннотации, введения или заключения, как правило, лучше отражают важные идеи научной статьи, тогда как предложения из разделов методологии, экспериментов или обзора литературы чаще содержат менее значимые детали. Поэтому при оценке значимости вершины используются два связанных, но не совпадающих признака: *секционный приоритет* и *взвешенная сумма опорных предложений*.

Для каждого предложения s сначала вычисляется секционный приоритет

$$\pi_{\text{sec}}(s) = \max\Big(0.6,\ \max_{\substack{k \in \mathcal{K} \\ k \subseteq \sigma(s)}} w_k\Big),$$

где $\sigma(s)$ обозначает нормализованное имя раздела (секции), $\mathcal{K}$ — множество секционных шаблонов, а w_k — соответствующие им весовые коэффициенты. Такая конструкция означает, что предложению назначается приоритет в зависимости от того, к какому типу секции оно принадлежит. Повышенные значения получают предложения из `Abstract`, `Conclusion` и `Introduction`, тогда как для разделов, ориентированных на технические детали, используются более низкие веса.

Если предложение s ранее было помечено как ссылочно-нагруженное, его секционный приоритет дополнительно штрафуется:

$$\pi'_{\text{sec}}(s) = 0.3 \cdot \pi_{\text{sec}}(s).$$

На уровне вершины секционный приоритет определяется усреднением по всем его опорным предложениям:

$$p_{\text{sec}}(v) = \frac{1}{|\mathcal{E}(v)|} \sum_{s \in \mathcal{E}(v)} \pi'_{\text{sec}}(s).$$

Следовательно, величина $p_{\text{sec}}(v)$ характеризует не объем поддержки вершины, а *среднее качество ее опорных предложений*. Если вершина опирается преимущественно на предложения из аннотации, введения и заключения, то ее $p_{\text{sec}}(v)$ будет высоким.

Наряду с этим вводится второй признак — взвешенная сумма опорных предложений:

$$c_{\text{ev}}(v) = \sum_{s \in \mathcal{E}(v)} w_{\text{ev}}(s),$$

где

$$w_{\text{ev}}(s) = \begin{cases} 1.5, & \text{если } s \text{ принадлежит секции } \texttt{Abstract}, \texttt{Conclusion} \\ 1.0, & \text{иначе.} \end{cases}$$

В отличие от $p_{\text{sec}}(v)$, величина $c_{\text{ev}}(v)$ отражает уже не среднее качество, а *объем текстовой поддержки вершины*. Она возрастает как при увеличении числа опорных предложений, так и при смещении их распределения в сторону наиболее информативных разделов.

Таким образом, два признака выполняют разные функции. Признак $p_{\text{sec}}(v)$ отвечает на вопрос: из каких по значимости секций в среднем получены доказательства для вершины v? Признак же $c_{\text{ev}}(v)$ отвечает на другой вопрос: насколько широко и насколько «сильно» данная вершина поддержана текстом документа? Использование обоих признаков одновременно позволяет различать, с одной стороны, вершины, подтвержденные большим числом предложений, а с другой — вершины, чья поддержка сосредоточена в наиболее концептуально значимых частях статьи.

Величины $p_{\text{sec}}(v)$ и $c_{\text{ev}}(v)$ нормируются к общему диапазону значений и затем входят в итоговую линейную комбинацию параметров значимости вершин.

Признаки автореферирования и общности вершин. Помимо структурных характеристик графа и признаков, связанных с опорными предложениями, для каждой вершины v вычисляются еще два дополнительных признака: *автореферирование* и *общность*.

Первый признак служит для выделения тех вершин, чьи опорные предложения содержат автореферентные формулировки, например,

“this paper”, “we propose”, “our method”, “we present”, “our approach”,

характерные для изложения собственного вклада статьи:

$$r_{\text{self}}(v) = \begin{cases} 1, & \exists s \in \mathcal{E}(v),\ \exists m \in \mathcal{M}_{\text{self}} :\ m \subseteq \text{lower}(s), \\ 0, & \text{иначе,} \end{cases}$$

где $\mathcal{M}_{\text{self}}$ — фиксированное множество автореферентных маркеров, а $\text{lower}(s)$ есть текст предложения в нижнем регистре.

После вычисления бинарных значений $r_{\text{self}}(v)$ для всех вершин они, как и остальные признаки, подвергаются общей min–max-нормировке.

Второй признак оценивает насколько метка вершины является общей или, напротив, узкой и специфичной. Важно подчеркнуть, что в рассматриваемой реализации общность метки вершины вычисляется не на основе внешней онтологии, а с помощью TF-IDF, и потом нормируются.

Если метка вершины содержит редкие токены, встречающиеся лишь в немногих вершинах графа, то *специфичность* будет большой. Если же токены метки распространены по многим вершинам, то специфичность будет ниже.

Итоговая значимость вершины. Значимость вершины определяется линейной комбинацией

$$\text{Sal}(v) = w_d\widehat{d}(v)+w_b\widehat{b}(v)+w_e hatd(v)+w_b\widehat{b}(v\widehat{c}_{\text{ev}}(v)+w_s\widehat{p}_{\text{sec}}(v)+w_g g(v)+w_r\widehat{r}_{\text{self}}(v).$$

В эталонной конфигурации используются веса

$$w_d = 0.15, \quad w_b = 0.10, \quad w_e = 0.30, \quad w_s = 0.25, \quad w_g = 0.10, \quad w_r = 0.10.$$

Построение рабочего графа G^*. Рабочий граф G^* строится из объединенного графа отбором наиболее значимых вершин:

$$V^* = \{v \in V : \text{Sal}(v) \geq \tau_{\text{node}}\}, \qquad \tau_{\text{node}} = 0.12,$$

с последующим ограничением сверху

$$|V^*| \leq N_{\max}, \qquad N_{\max} = 200.$$

Если после применения порогового ограниения множество пусто, сохраняется хотя бы одна наиболее значимая вершина.

Дополнительно даже если некоторые вершины не прошли по порогу, в V^* принудительно включается не менее $n_{\text{abs}} = 2$ вершин, поддержанных опорными предложениями из аннотации, и не менее $n_{\text{conc}} = 1$ вершины из заключения.

Для каждого ребра $e = (u, r, v)$ с обоими концами в V^* вычисляется значимость

$$\text{Sal}_E(e) = 0.45 \cdot \frac{\text{Sal}(u) + \text{Sal}(v)}{2} + 0.35 \cdot \mathbf{1}[c_e > 0] + 0.20 \cdot q(r),$$

где $q(r)$ — априорное качество типа отношения:

$$q(r) = \begin{cases} 1.00, & r = \mathsf{part\_of}, \\ 0.95, & r = \mathsf{feature\_of}, \\ 0.90, & r = \mathsf{used\_for}, \\ 0.85, & r = \mathsf{hyponym\_of}, \\ 0.80, & r = \mathsf{evaluate\_for}, \\ 0.65, & r = \mathsf{compare}, \\ 0.55, & r = \mathsf{conjunction}, \\ 0.70, & \text{иначе.} \end{cases}$$

Если число ребер превышает лимит

$$|E^*| > E_{\max}, \qquad E_{\max} = 420,$$

то сохраняются только ребра с наибольшими значениями Sal_E, а вершины не удаляются даже в случае потери всех инцидентных ребер.

3.6 Поиск сообществ и построение иерархии в графе

Вложенные сообщества на графе G^*. После построения рабочего графа

$$G^* = (V^*, E^*)$$

выполняется группировка его вершин в крупные тематические области, называемые фасетами. Для этого из G^* строится простая неориентированная проекция

$$U_{\text{facet}} = (V_{\text{facet}}, E_{\text{facet}}), \qquad V_{\text{facet}} = V^*.$$

Для каждой пары вершин $u, v \in V^*$ вес неориентированного ребра определяется как максимальная значимость среди всех дуг, соединяющих эти узлы в G^*:

$$w_{\text{facet}}(u, v) = \max_{e \in \mathcal{P}(u,v)} \mathrm{Sal}_E(e),$$

где $\mathcal{P}(u, v)$ обозначает множество всех ребер между u и v без учета ориентации. Использование максимума, а не суммы или среднего, позволяет сохранить для каждой пары вершин наиболее сильный тематический сигнал связи.

На графе U_{facet} далее выполняется поиск сообществ:

$$\mathrm{CommAlg}(U, \gamma),$$

где γ — параметр, управляющий размером сообществ.

Поиск сообществ выполняется с помощью алгоритма Leiden [48], который был выбран по результатам экспериментов.

Операция вложенного разбиения графа на кластеры:

$$\mathcal{C}^{(1)} = \{C_1^{(1)}, \ldots, C_{m_1}^{(1)}\} = \mathrm{CommAlg}(U_{\mathrm{facet}}, \gamma_1).$$

Для каждого сообщества верхнего уровня $C_i^{(1)}$ при достаточном размере строится индуцированный подграф

$$U_{\mathrm{facet}}[C_i^{(1)}],$$

после чего внутри него повторно запускается поиск сообществ:

$$\mathcal{C}_i^{(2)} = \mathrm{CommAlg}\big(U_{\mathrm{facet}}[C_i^{(1)}], \gamma_2\big).$$

В конфигурационном файле предусмотрена настройка числа уровней сообществ от 1 до 3. Наилучший результат по итогам экспериментов получило значение 2.

Таким образом, создается иерархия сообществ

$$L1 \;\rightarrow\; L2 \;\rightarrow\; L3,$$

в которой уровень $L1$ соответствует крупным темам документа, уровень $L2$ — подтемам, а уровень $L3$ — более мелким локальным кластерам. Углубление выполняется только при выполнении трех условий: (i) текущее сообщество содержит не менее заданного минимального числа вершин, (ii) его индуцированный подграф не пуст по ребрам, (iii) повторный запуск выявления сообществ действительно разбивает его более чем на одно подсообщество. Если хотя бы одно из этих условий нарушается, рекурсия останавливается, и соответствующее сообщество считается листом иерархии.

После построения сообществ верхнего уровня для них дополнительно вычисляется суммарный показатель важности:

$$\mathrm{CommSal}(C) = \sum_{v \in C} \mathrm{Sal}(v),$$

после чего верхнеуровневые сообщества упорядочиваются по убыванию $\mathrm{CommSal}(C)$. При необходимости число таких сообществ ограничивается сверху параметром `max_facets`, так что в дальнейшее построение иерархической сводки попадают только наиболее значимые крупные темы.

Содержательно данный этап решает две задачи. Во-первых, он разбивает рабочий граф на крупные тематические области, что позволяет избежать смешивания различных аспектов статьи в одной ветке иерархии. Во-вторых, вложенная структура сообществ создает естественный каркас для промежуточных уровней иерархии.

Иерархический граф по типизированным отношениям. Параллельно с выделением сообществ из G^* извлекается ориентированный граф отношений типа «родитель–потомок» (HierarchyDAG). Для этого используются отношения

$$\mathcal{R}_{\text{hier}} = \{\mathsf{part\_of}, \mathsf{hyponym\_of}, \mathsf{feature\_of}\}.$$

Каждая такая дуга интерпретируется как сигнал отношения «более частное $\to$ более общее». Для каждой вершины x выбирается не более одного родителя:

$$\text{Parent}_{\text{hier}}(x) = \arg\max_{p \in \Pi(x)} \text{Sal}_E(x \to p),$$

где $\Pi(x)$ — множество допустимых родителей по типизированным ребрам.

Если выбор очередного родителя создает цикл, такой кандидат отбрасывается и рассматривается следующий по убыванию Sal_E. В результате формируется ациклический граф

$$\mathcal{H} = (V^*, E_{\mathcal{H}}).$$

Вместе с тем, для каждого верхнеуровнего сообщества $C \in \mathcal{C}^{(1)}$ выбирается корень темы

$$r_C = \arg\max_{v \in C} \big(g(v), \text{Sal}(v)\big),$$

где $g(v)$ — показатель общности вершины, а $\text{Sal}(v)$ — его общая значимость. При прочих равных предпочтение отдается вершинам, не имеющим родителя внутри данного сообщества по иерархическому графу.

Если внутри сообщества верхнего уровня имеются содержательные сообщества второго уровня, то для каждого из них аналогичным образом выбирается представитель, который становится промежуточной вершиной дерева и задает дополнительный уровень детализации внутри крупной темы. После выбора корней темы и представителей ее подсообществ прикрепляются остальные вершины.

Для сохранения читаемости иерархии дополнительно вводятся ограничения на максимальную глубину дерева и максимальное число дочерних вершин на каждом уровне. Поэтому получаемая структура остается деревом в графовом смысле: каждая вершина, кроме корня, имеет единственного родителя, а сама сводка читается как последовательное раскрытие тем от более общих к более частным.

3.7 Сопоставление вершинам дерева опорных предложений

После того как на графовом уровне построено дерево проекта иерархической сводки, каждой его вершине необходимо сопоставить одно опорное предложение, которая будет отображаться в итоговой сводке. Этот этап является *экстрактивным*: текст вершины выбирается из исходных предложений документа, а не генерируется заново. Тем самым сохраняется прямая привязка каждого пункта сводки к исходному тексту статьи.

Обозначим вершину графа $\ell(v)$, а через

$$\mathcal{E}(v) = \{s_1, \ldots, s_m\}$$

обозначим множество ее собственных опорных предложений. Кроме того, для вершины v можно рассмотреть опорные предложения ее поддерева:

$$\mathcal{E}_{\downarrow}(v) = \mathcal{E}(v) \cup \bigcup_{u \in \mathrm{Desc}(v)} \mathcal{E}(u).$$

Тогда пул предложений-кандидатов для присвоения вершине формируется как упорядоченная последовательность

$$\mathcal{C}(v) = \mathcal{E}(v) \parallel \mathcal{E}_{\downarrow}(v),$$

где символ $\parallel$ означает конкатенацию с приоритетом собственных опорных предложений вершины перед опорными предложениями ее потомков: сначала предпочтение отдается локальным предложениям, непосредственно связанным с данной вершиной, а уже затем, при необходимости, допускается выбор предложения из ее поддерева.

Для каждого предложения-кандидата $s \in \mathcal{C}(v)$ вычисляется показатель согласованности с меткой вершины. Пусть

$$T(v) = \mathrm{Tok}(\ell(v)), \qquad T(s) = \mathrm{Tok}(s),$$

где $\mathrm{Tok}(\cdot)$ обозначает множество словарных токенов после приведения к нижнему регистру. Тогда базовый вклад лексического совпадения опре-

деляется как доля токенов метки вершины, найденных в тексте предложения:

$$\mathrm{Overlap}(v, s) = \frac{|T(v) \cap T(s)|}{|T(v)| + \varepsilon}.$$

Дополнительно учитывается секционный приоритет предложения:

$$\pi_{\mathrm{sec}}(s),$$

поэтому итоговая оценка кандидата имеет вид

$$\chi(s \mid v) = \mathrm{Overlap}(v, s) + 0.2 \cdot \pi_{\mathrm{sec}}(s).$$

Таким образом, предпочтение отдается предложениям, которые, с одной стороны, лексически лучше согласуются с меткой вершины графа, а с другой — происходят из более информативных разделов статьи, таких как `Abstract` или `Conclusion`.

Вместе с тем, одной только оценки $\chi(s \mid v)$ недостаточно, поскольку при выборе максимально информативные предложения легко начинают повторяться вдоль одной и той же ветви дерева. Поэтому перед выбором вводятся два множества ограничений:

$$B_{\mathrm{id}}(v) \subseteq \mathcal{S}, \qquad B_{\mathrm{text}}(v) \subseteq \mathcal{Y},$$

где $B_{\mathrm{id}}(v)$ содержит идентификаторы предложений, уже использованных выше по текущей ветви, а $B_{\mathrm{text}}(v)$ — нормализованные тексты уже выбранных формулировок предков. Нормализация текста задается функцией

$$\nu(y) = \mathrm{lower}\big(\mathrm{collapse\_spaces}(y)\big).$$

Предложение-кандидат s считается запрещенным на данном шаге, если

$$s \in B_{\mathrm{id}}(v) \quad \text{или} \quad \nu(s) \in B_{\mathrm{text}}(v).$$

Тем самым алгоритм старается не только избегать повторного использования одного и того же sentence_id, но и предотвращать почти дословное повторение одной и той же формулировки в нескольких соседних уровнях дерева.

3.8 Итеративное уточнение иерархии с помощью БЯМ

На данном этапе алгоритм получает на вход уже построенное графовое дерево

$$\mathcal{T}_{\mathrm{graph}}.$$

Это является принципиальным отличием от прямой генерации сводки по исходному тексту: модель работает не с сырым полным текстом статьи, а с уже подготовленным графовым каркасом, содержащим верхние ветви, наборы опорных предложений и подтемы-кандидаты. Тем самым БЯМ не строит иерархию *с нуля*, а уточняет и переформулирует уже существующую структуру, построенную детерминированным графовым этапом.

Содержательно этот этап решает две задачи. Во-первых, он делает верхний уровень сводки более связным и читабельным: вместо механического списка вершин графа БЯМ выбирает ограниченное число верхнеуровневых ветвей, которые лучше складываются в повествовательную структуру сводки. Во-вторых, он рекурсивно углубляет отдельные ветви, если для этого есть достаточное число опорных предложений, формируя более детализированное дерево. Все ответы модели после генерации проходят жесткую проверку по допустимым идентификаторам предложений, опорным предложениям и затем «обрезку» ветвей иерархии.

Верхний уровень. Для корня дерева формируется множество верхнеуровневых тем-кандидатов

$$\mathcal{Q}_{\text{top}} = \text{Top}(\mathcal{T}_{\text{graph}}),$$

которое фактически представляет собой плоский список ближайших графовых подтем корня с их опорными предложениями. Параллельно собирается набор опорных предложений для верхнего уровня

$$\mathcal{P}_{\text{top}} = \text{Evidence}(r), \qquad |\mathcal{P}_{\text{top}}| \leq 160.$$

В набор попадают уникальные предложения из поддерева корня, отсортированные по порядку появления в статье. Такое ограничение необходимо для контроля длины контекста и стоимости запроса.

В БЯМ на верхнем уровне передаются:

1. заголовок статьи или корпуса;
2. список тем-кандидатов;
3. набор опорных предложений;
4. пустой список уже покрытых формулировок («already_covered»);
5. требуемый диапазон числа верхнеуровневых ветвей.

Модель должна вернуть от 4 до 8 верхнеуровневых ветвей. Каждая ветвь обязана иметь структуру

$$(\text{sentence},\ \text{evidence\_sentence\_ids},\ \text{children}),$$

где поле «sentence» содержит короткую однофразовую формулировку темы, «evidence_sentence_ids» — непустой набор ссылок на предложения из допустимого набора, а «children» на этом шаге либо пуст, либо служит контейнером для дальнейшего рекурсивного раскрытия. Модель также получает инструкцию не повторять заголовок статьи и не строить верхний уровень как набор почти одинаковых ветвей.

После получения ответа БЯМ верхнеуровневые кандидаты проходят первичную очистку. Безусловно отбрасываются ветви, начинающиеся с анафорических слов вида “it”, “this”, “they”, поскольку такие формулировки плохо читаются как самостоятельные верхнеуровневые пункты. Затем отбрасываются близкие к дубликатам ветви на основании измерения сходства по Жаккару.

Рекурсивное раскрытие ветвей. После предыдущего шага каждая верхнеуровневая ветвь раскрывается рекурсивно. Для текущей вершины u, находящейся на глубине d, рассматриваются:

$$\mathcal{Q}(u) = \text{набор графовых подтем-кандидатов из соответствующей ветви,}$$

$$\mathcal{P}(u) = \text{набор опорных предложений, собранный из поддерева } u.$$

Набор опорных предложений данного уровня ограничивается сверху:

$$|\mathcal{P}(u)| \le 80,$$

а число подтем-кандидатов также ограничивается отдельным бюджетом.

Рекурсивное расширение ветви выполняется, если соблюдено хотя бы одно из двух условий:

$$d < D_{\min}^{\mathrm{LLM}} \quad \text{и} \quad \big(\mathcal{Q}(u) \neq \varnothing \ \vee\ |\mathcal{P}(u)| \ge 1\big),$$

или

$$d < D_{\max}^{\mathrm{LLM}} \quad \text{и} \quad |\mathcal{Q}(u)| \ge C_{\min} \quad \text{и} \quad |\mathcal{P}(u)| \ge E_{\min},$$

где по умолчанию

$$D_{\min}^{\mathrm{LLM}} = 5, \qquad D_{\max}^{\mathrm{LLM}} = 8, \qquad C_{\min} = 3, \qquad E_{\min} = 2.$$

Первое условие задает режим *быстрого углубления*: пока глубина дерева меньше минимально желаемой, алгоритм старается продолжать детализацию, если у ветви есть хотя бы какие-то структурные сигналы. Второе условие задает обычный режим после достижения минимальной глубины: дальнейшее углубление разрешено только тогда, когда у вершины есть достаточное число потомков-кандидатов и достаточный набор опорных предложений. Тем самым система избегает как преждевременной остановки слишком мелкого дерева, так и бессодержательного расширения ветвей при нехватке исходного материала.

Локальная дедупликация и фильтрация ветвей-потомков. В рамках данного шага безусловно удаляются анафорические формулировки, начинающиеся с местоимений типа «It» и подобных. Также отбрасываются вершины, являющиеся почти дубликатом своего родителя или любого предка в текущей цепочке или любой уже принятой вершины в других ветвях дерева. Для этого используется сходство по Жаккару над содержательными токенами с порогом 0.7. Внутри одного вызова дополнительно ограничивается число соседних вершин с одинаковым первым словом: если их оказывается больше двух, лишние узлы отбрасываются. В результате на выходе данного шага остаются только новые, достаточно отличающиеся друг от друга и читаемые вершины.

Финальная «обрезка» дерева. После окончания рекурсии выполняется дополнительная глобальная проверка. Для каждой вершины u проверка опорных предложений осуществляется уже не только по его собственному их набору, но и по объединению наборов опорных предложений всех предков:

$$\mathcal{E}^{+}(u) = \mathcal{E}(u) \cup \bigcup_{a \in \mathrm{Anc}(u)} \mathcal{E}(a).$$

Смысл данного шага состоит в том, что дочерняя вершина может быть корректно понята только на фоне родительского контекста. Если вершина не подтверждается объединенным набором опорных предложений, то соответствующее поддерево удаляется. Такая процедура особенно полезна для тех случаев, когда локально вершина выглядит правдоподобно, но после встраивания в общую иерархию оказывается недостаточно сопоставленной со смыслом контекста.

3.9 Дедупликация и удаление «плохих» фрагментов

Нормализация текста. Для текста вершины y введем нормализованную форму

$$\nu(y) = \mathrm{lower}\big(\mathrm{collapse\_spaces}(y)\big),$$

а для пары вершин u, v определим меру похожести по Жаккару:

$$\mathrm{ND}(u,v) = 1 \iff \frac{|\,\mathrm{Tok}(y(u)) \cap \mathrm{Tok}(y(v))|}{|\,\mathrm{Tok}(y(u)) \cup \mathrm{Tok}(y(v))|} \geq \tau_{\mathrm{dup}}, \qquad \tau_{\mathrm{dup}} = 0.92.$$

Если у родителя и ребенка совпадают нормализованные тексты,

$$\nu(y(parent)) = \nu(y(child)),$$

то ребенок удаляется, его опорные предложения вливаются в набор родителя, а его дети поднимаются на уровень выше.

Плохие фрагменты. Вершина считается плохим фрагментом, если выполнено условие $\mathrm{Bad}(y) = 1$, где Bad активируется, например, при следующих ситуациях:

1. слишком короткий фрагмент:

$$|\,\mathrm{Tok}(y)| \leq 2;$$

2. короткая анафорическая формулировка вида «this method», «it works»;
3. прямые ссылки на таблицы и рисунки;
4. строки, преимущественно состоящие из перечисления метрик или показателей;
5. слишком низкая доля буквенных символов (доля меньше 0.45).

Если $\mathrm{Bad}(y) = 1$ и у вершины есть дочерние, то сама вершина удаляется, а поддерево поднимается выше по иерархии.

3.10 Итоговое представление иерархической сводки

На выходе алгоритма формируется дерево

$$\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, r),$$

в котором:

1. корень соответствует заголовку статьи или корпуса;
2. прямые дети корня — верхнеуровневые темы (наиболее важные идеи, концепции научной статьи);
3. глубина дерева отражает переход от общих тем к деталям.

Таким образом, итоговая сводка не является ни чисто экстрактивным списком предложений, ни полностью свободно сгенерированным текстом. Она представляет собой структуру, построенную по графу, но допускающую переформулирование текстов вершин с помощью БЯМ при обязательной опоре на исходные предложения.

4 Метрики и оценка качества

В задачах суммаризации текстов вопрос критериев оценки или метрик является одним из самых важных. Целью любого подхода является минимизация различия между сгенерированными алгоритмом сводками и «золотым стандартом» [13]. Действительно, каким образом отличить две сгенерированные автоматическими алгоритмами иерархические сводки?

Некоторые авторы отмечают, что в области автоматической суммаризации текстов отсутствует единый, современный и полный подход к оценке моделей и метрик [7].

ROUGE (Recall-Oriented Understudy for Gisting Evaluation). Это набор метрик для автоматической оценки качества суммаризации, основанный на подсчёте пересечения n-грамм (последовательностей слов) между сгенерированным текстом и одним или несколькими эталонными сводками [49]. Основной упор делается на полноту (recall), то есть долю эталонных n-грамм, которые присутствуют в сводке.

Основные различия между вариантами метрики ROUGE заключаются в типе сравниваемых текстовых единиц: ROUGE-1 оценивает пересечение по униграммам (отдельным словам), полностью игнорируя порядок слов и фактически рассматривая текст как «мешок слов». ROUGE-2 использует биграммы (пары идущих подряд слов), что делает его чувствительным к локальному порядку и фразеологии, но не к глобальной структуре всего предложения. ROUGE-L основан на поиске наибольшей общей подпоследовательности и учитывает глобальный порядок слов, допуская при этом разрывы в последовательности (в отличие от ROUGE-2,

где биграммы должны быть строго непрерывными).

ROUGE продолжает широко использоваться благодаря своей простоте, интерпретируемости и дешевизне вычислений . Однако современные исследования выявили ряд серьезных ограничений, которые важно учитывать [7]:

- нечувствительность к синонимии и перефразированию: ROUGE считает совпадения только на уровне слов. Если в эталоне слово «собака», а в сгенерированной сводке - «пес», ROUGE не засчитает совпадение, хотя смысл тот же;
- плохая корреляция с человеческими оценками для некоторых языков с богатой морфологией (например, русский, немецкий);
- в задачах суммаризации научных текстов ROUGE может оценить изменение как ухудшение, даже если оно объективно улучшает ясность и стиль, так как метрика заточена на близость к единственной версии, а не на абсолютное качество.

Ряд авторов также указали на необходимость разработки новых метрик оценки качества аннотаций, поскольку такие подходы как ROUGE устарели и больше не соответствуют современным требованиям [21].

Тем не менее, представляется, что лексические метрики семейства ROUGE, хотя и признаются устаревшими многими исследователями, все-таки могут дать критерии для сравнения отдельных элементов или уровней иерархии. Эти показатели полезны, но ограничены в задачах иерархической суммаризации, поскольку практически не чувствительны к корректности дерева как структуры.

Однако, основными метриками, используемыми для оценки, являются: TTED, STTED, BERTScore.

TTED. Метрика TTED (*Text Tree Edit Distance*) служит основной структурно-семантической мерой различия между двумя текстовыми деревьями. По своей идее она восходит к классическому расстоянию редактирования деревьев, однако стоимость операций редактирования определяется не только структурой, но и семантикой текстов в вершинах, что впервые предложено Ф.А. Соболевским [50]. Иначе говоря, если в обычном tree edit distance замена узла трактуется как дискретная операция, то в TTED ее «цена» зависит от семантической близости текстов в соответствующих вершинах.

На практике это означает, что:

- стоимость замены вершины u на вершину v определяется семантической дистанцией между их текстами;
- стоимость вставки и удаления вершины задаётся как расстояние между текстом вершины и “пустым” текстом;
- итоговая величина отражает одновременно различия в составе тем, их формулировках и иерархическом расположении.

Чем меньше значение TTED, тем ближе кандидат к эталону. В приложении эта метрика вычисляется в нормализованном режиме и в упорядоченной постановке, то есть порядок дочерних узлов учитывается.

STTED. Метрика STTED используется как вторая основная древовидная дистанция и рассматривается как структурно ориентированная дистанционная метрика и агрегируется так же, как TTED: меньшие значения соответствуют лучшему совпадению кандидата с эталоном. С практической точки зрения STTED удобно интерпретировать как дополнительную модификацию древовидного расстояния, позволяющую проверить устойчивость выводов не к одной, а к двум близким, но не тождественным структурным мерам.

BERTScore. Метрика BERTScore [51] используется как основная семантическая мера текстового сходства. В отличие от лексических метрик она сравнивает кандидата и эталонную сводку в пространстве контекстных эмбеддингов и потому лучше учитывает перефразирование и семантически близкие формулировки. Для задач иерархической суммаризации это особенно важно, поскольку корректная тема может быть выражена иными словами, чем в «золотом стандарте».

Чем выше BERTScore, тем лучше совпадение по смыслу.

Интерпретация метрик в задаче иерархической суммаризации. Таким образом, набор используемых метрик покрывает различные аспекты качества:

1. **семантическое совпадение содержания** (BERTScore, частично ROUGE);
2. **структурное совпадение дерева тем и подтем** (TTED, STTED.

Такой подход предпочтительнее использования одной плоской текстовой метрики, поскольку иерархическая сводка должна быть оценена не только по совпадению слов, но и по правильности организации содержания: какие темы выделены, как они сгруппированы и на каком уровне иерархии расположены.

Подробнее с методикой и результатами сравнения можно ознакомиться в Приложение В. Сравнение иерархических сводок, созданных различными алгоритмами и системами с «золотым стандартом»

5 Научная новизна и отличие от похожих методов

Предлагаемый подход отличается от классических графовых методов ранжирования наподобие LexRank [52] тем, что граф используется не только для оценки центральности предложений. В нем формируется явно типизированное промежуточное представление в виде локальных миниграфов, далее агрегируемых в единый мультиграф. Значимость вершин определяется не одной центральностью, а комбинацией структурных, секционных, опорных и лексических признаков. Следовательно, итоговая иерархия возникает не как список «самых важных» предложений, а как композиция тематических сообществ, направленных связей и текстовых единиц из исходной статьи.

От графовых нейросетевых методов суммаризации [53], в частности от семейства, в котором граф используется как вход для обучаемого encoder–decoder или sequence labeling, предлагаемый метод отличается отсутствием end-to-end обучения под конкретный датасет. Здесь графовый каркас строится детерминированно, а БЯМ применяется как внешний модуль только на этапах извлечения миниграфа и/или уточнения уже построенной иерархии. Это делает метод более интерпретируемым: можно отдельно анализировать, какая вершина была включена в рабочий граф, почему она оказалась значимой, в какое сообщество попала и какими опорными предложениями поддерживается. Кроме того, такой метод обладает гораздо большей обобщающей способностью, нежели обученная на конкретном датасете модель.

От иерархических моделей архитектуры «трансформер» [54] предлагаемый подход отличается тем, что не кодирует весь корпус одной нейронной сетью. Вместо этого сначала строится разреженная структура в виде графа, а затем уже по ней восстанавливается иерархия. Такой дизайн особенно полезен там, где важны проверяемость и возможность

исследовать промежуточные артефакты на всех шагах алгортима.

Наконец, от GraphRAG-подобных алгоритмов [55] [56] метод отличается целевой постановкой задачи применения графа. В GraphRAG граф обычно служит индексом для ответа на внешний запрос, а в рассмотренном методе — *центральный объект построения самой иерархической сводки.* Иными словами, результатом является не ответ на вопрос поверх графа, а само дерево тем, подтем и деталей, связанное с исходным текстом через опорные предложения.

С научной точки зрения ключевая новизна метода состоит в следующем:

1. используется трехуровневая схема «локальные миниграфы → глобальный рабочий граф → иерархическое дерево»;

2. объединяются два независимых механизма построения иерархии: выделение сообществ и иерархический граф по типизированным отношениям;

3. БЯМ встроена не как единственный генератор итогового текста, а как контролируемый модуль улучшения и коррекции поверх детерминированного графового каркаса;

4. для всех вершин после обработки БЯМ реализована обязательная проверка на наличие опорных предложений, что снижает риск необоснованных верхнеуровневых или промежуточных тематик в итоговой иерархической сводке.

Список литературы

[1] Yang Christopher C., Wang Fu Lee. Hierarchical Summarization of Large Documents // Journal of the American Society for Information Science and Technology. — 2008. — Vol. 59, no. 6. — P. 887–902.

[2] Hierarchical Summarization: Scaling Up Multi-Document Summarization / Christensen Janara, Soderland Stephen, Bansal Gagan, and Mausam // Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Baltimore, Maryland : Association for Computational Linguistics. — 2014. — P. 902–912.

[3] Бьюзен Тони. Интеллект-Карты. Полное Руководство По Мощному Инструменту Мышления. — Москва : Манн, Иванов и Фербер, 2019.

[4] Chen Chaomei. Mapping Scientific Frontiers: The Quest for Knowledge Visualization. — Second Edition ed. — London : Springer, 2013.

[5] Константин Воронцов. Карты знаний. На пути к доверенным языковым моделям и системам представления знаний. — Режим доступа: `https://ib-bank.ru/bisjournal/post/2290` (дата обращения: 2025-09-08).

[6] Гибридный метод автореферирования научно-технических текстов на основе риторического анализа / Батура Т.В., Batura T.V., Бакиева А.М. и A.M. Bakieva // Международный журнал "Программные продукты и системы". — 2020. — Т. 21. — С. 144–153.

[7] SummEval: Re-evaluating Summarization Evaluation / Fabbri Alexander R., Kryściński Wojciech, McCann Bryan, Xiong Caiming, Socher Richard, and Radev Dragomir // Transactions of the Association for Computational Linguistics. — 2021. — Vol. 9. — P. 391–409.

[8] Dang Hoa Trang. Information Access Division National Institute of Standards and Technology Gaithersburg, MD 20899 // DUC 2005. — 2005.

[9] Rezapour-Nasradad Rafat. Mind Map Learning Technique: An Educational Interactive Approach // International Journal of Pharmaceutical Research. — 2019. — P. Vol 11.

[10] Towards an AI-Augmented Textbook. — 2025. — arXiv : 2509.13348.

[11] Jose Guerrero. Mind Mapping for Reading and Understanding Scientific Literature // International Journal of Current Advanced Research. — 2015. — P. 485–487.

[12] From Tradition to Innovation: Mind Map Generation in Higher Education / Mitra Aditya Rama, Samosir Feliks Victor Parningotan, Hudi Robertus, and Tarigan Riswan Effendi // Ultima InfoSys : Jurnal Ilmu Sistem Informasi. — 2023. — Vol. 14, no. 2. — P. 71–78.

[13] Multi-Document Summarization via Deep Learning Techniques: A Survey. — 2021. — arXiv : cs/2011.04843.

[14] Li Miao, Hovy Eduard, Lau Jey Han. Summarizing Multiple Documents with Conversational Structure for Meta-Review Generation. — 2023. — arXiv : cs/2305.01498.

[15] ScisummNet: A Large Annotated Corpus and Content-Impact Models for Scientific Paper Summarization with Citation Networks. — 2019. — arXiv : cs/1909.01716.

[16] Luhn H. P. The Automatic Creation of Literature Abstracts // IBM Journal of Research and Development. — 1958. — Vol. 2, no. 2. — P. 159–165.

[17] Direct Automatic Generation of Mind Maps from Text with M^2Gen / Abdeen M., El-Sahan R., Ismaeil A., El-Harouny S., Shalaby M., and Yagoub M. C.E. // 2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH). — Toronto, ON, Canada : IEEE. — 2009. — P. 95–99.

[18] Elhoseiny Mohamed, Elgammal Ahmed. English2MindMap: An Automated System for MindMap Generation from English Text // 2012 IEEE International Symposium on Multimedia. — Irvine, CA, USA : IEEE. — 2012. — P. 326–331.

[19] A Study of Extractive Summarization of Long Documents Incorporating Local Topic and Hierarchical Information / Wang Ting, Yang Chuan, Zou Maoyang, Liang Jiaying, Xiang Dong, Yang Wenjie, Wang Hongyang, and Li Jia // Scientific Reports. — 2024. — Vol. 14, no. 1. — P. 10140.

[20] Ou Litu, Lapata Mirella. Context-Aware Hierarchical Merging for Long Document Summarization. — 2025. — arXiv : cs/2502.00977.

[21] Pu Xiao, Gao Mingqi, Wan Xiaojun. Summarization Is (Almost) Dead. — 2023. — arXiv : cs/2309.09558.

[22] Mixture of Knowledge Minigraph Agents for Literature Review Generation. — 2025. — arXiv : cs/2411.06159.

[23] Leveraging Graph to Improve Abstractive Multi-Document Summarization / Li Wei, Xiao Xinyan, Liu Jiachen, Wu Hua, Wang Haifeng, and Du Junping // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. — Online : Association for Computational Linguistics. — 2020. — P. 6232–6243. — Access mode: https://www.aclweb.org/anthology/2020.acl-main.555.

[24] A Comprehensive Survey on Process-Oriented Automatic Text Summarization with Exploration of LLM-Based Methods. — 2025. — arXiv : cs/2403.02901.

[25] Planning with Learned Entity Prompts for Abstractive Summarization. — 2021. — arXiv : cs/2104.07606.

[26] Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. — 2023. — arXiv : cs/2201.11903.

[27] Leviathan Yaniv, Kalman Matan, Matias Yossi. Prompt Repetition Improves Non-Reasoning LLMs. — 2025. — arXiv : cs/2512.14982.

[28] Chen Xiaoyong, Chen Zhiqiang, Cheng Shi. CoTHSSum: Structured Long-Document Summarization via Chain-of-Thought Reasoning and Hierarchical Segmentation // Journal of King Saud University Computer and Information Sciences. — 2025. — Vol. 37, no. 4. — P. 40.

[29] Beyond Generic Summarization: A Multi-faceted Hierarchical Summarization Corpus of Large Heterogeneous Data / Tauchmann Christopher, Arnold Thomas, Hanselowski Andreas, Meyer Christian M., and Mieskes Margot // Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). — Miyazaki, Japan : European Language Resources Association (ELRA). — 2018. — May. — Access mode: https://aclanthology.org/L18-1503/.

[30] Towards Coherent Multi-Document Summarization / Christensen Janara, Mausam, Soderland Stephen, and Etzioni Oren // Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies / ed. by Vanderwende Lucy, Daumé III Hal, Kirchhoff Katrin. — Atlanta, Georgia : Association for Computational Linguistics. — 2013. — June. — P. 1163–1173. — Access mode: https://aclanthology.org/N13-1136/.

[31] Chan Joel. Discourse Graphs for Augmented Knowledge Synthesis: What and Why. — Access mode: https://joelchan.me/assets/pdf/Discourse_Graphs_for_Augmented_Knowledge_Synthesis_What_and_Why.pdf.

[32] Dr. Evan Miyazono Dr. Matt Akamatsu. Discourse graphs and the future of science. — Access mode: https://research.protocol.ai/blog/2023/discourse-graphs-and-the-future-of-science/.

[33] Mirasdar Sharayu, Mangesh Bedekar. Knowledge Graphs for NLP: A Comprehensive Analysis // The Scientific Temper. — 2025. — Vol. 16. — P. 141–148.

[34] Hogan Aidan, Blomqvist Eva, Cochez et al. Knowledge Graphs // ACM Comput. Surv. — 2021. — Mar.

[35] Krötzsch Markus, Weikum Gerhard. Editorial // Journal of Web Semantics. — 2016. — Vol. 37–38. — P. 53–54.

[36] Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO / Färber Michael, Bartscherer Frederic, Menne Carsten, and Rettinger Achim // Semantic Web. — 2017. — Vol. 9, no. 1. — P. 77–129.

[37] Ehrlinger Lisa, Wöß Wolfram. Towards a Definition of Knowledge Graphs // Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change and Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016). — Leipzig, Germany. — 2016. — Sep. — Access mode: https://ceur-ws.org/Vol-1695/paper4.pdf.

[38] Graph-Based Neural Multi-Document Summarization. — 2017. — arXiv : cs/1706.06681.

[39] Efficient Mind-Map Generation via Sequence-to-Graph and Reinforced Graph Refinement. — 2021. — arXiv : cs/2109.02457.

[40] Multi-Document Scientific Summarization from a Knowledge Graph-Centric View / Wang Pancheng, Li Shasha, Pang Kunyuan, He Liangliang, Li Dong, Tang Jintao, and Wang Ting // Proceedings of the 29th International Conference on Computational Linguistics. — Gyeongju, Republic of Korea : International Committee on Computational Linguistics. — 2022. — . — P. 6222–6233. — Access mode: https://aclanthology.org/2022.coling-1.543/.

[41] Coreference Graph Guidance for Mind-Map Generation. — 2023. — arXiv : cs/2312.11997.

[42] Lu Jiaying, Dong Xiangjue, Yang Carl. Weakly Supervised Concept Map Generation Through Task-Guided Graph Translation // IEEE Transactions on Knowledge and Data Engineering. — 2023. — Vol. 35, no. 10. — P. 10871–10883. — Access mode: https://ieeexplore.ieee.org/document/10059210/.

[43] Kipf Thomas N., Welling Max. Semi-Supervised Classification with Graph Convolutional Networks. — 2017. — arXiv : cs/1609.02907.

[44] CS-KG 2.0: A Large-scale Knowledge Graph of Computer Science / Dessí Danilo, Osborne Francesco, Buscaldi Davide, Reforgiato Recupero Diego, and Motta Enrico // Scientific Data. — 2025. — Vol. 12, no. 1. — P. 964.

[45] Romanova Alex. Enhancing NLP through GNN-Driven Knowledge Graph Rewiring and Document Classification // 2024 35th Conference of Open Innovations Association (FRUCT). — Tampere, Finland : IEEE. — 2024. — P. 579–587. — Access mode: https://ieeexplore.ieee.org/document/10516410/.

[46] Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction / Luan Yi, He Luheng, Ostendorf Mari, and Hajishirzi Hannaneh // Proceedings of the

2018 Conference on Empirical Methods in Natural Language Processing. — Brussels, Belgium : Association for Computational Linguistics. — 2018. — P. 3219–3232. — Access mode: http://aclweb.org/anthology/D18-1360.

[47] Angelov Dimo, Inkpen Diana. Topic Modeling: Contextual Token Embeddings Are All You Need // Findings of the Association for Computational Linguistics: EMNLP 2024. — Miami, Florida, USA : Association for Computational Linguistics. — 2024. — P. 13528–13539. — Access mode: https://aclanthology.org/2024.findings-emnlp.790.

[48] Traag Vincent, Waltman Ludo. From Louvain to Leiden: Guaranteeing Well-Connected Communities // Scientific Reports. — 2019. — Vol. 9, no. 1. — P. 5233. — arXiv : cs/1810.08473.

[49] Lin Chin-Yew. ROUGE: A Package for Automatic Evaluation of Summaries // Text Summarization Branches Out. — Barcelona, Spain : Association for Computational Linguistics. — 2004. — July. — P. 74–81. — Access mode: https://aclanthology.org/W04-1013/.

[50] Sobolevsky Fedor, Vorontsov Konstantin. Text Tree Edit Distance: A Language Model-Based Metric for Text Hierarchies // 2025 IEEE XVII International Scientific and Technical Conference on Actual Problems of Electronic Instrument Engineering (APEIE). — 2025. — P. 1–5.

[51] BERTScore: Evaluating Text Generation with BERT. — 2020. — arXiv : cs/1904.09675.

[52] Erkan Güneş, Radev Dragomir R. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization // Journal of Artificial Intelligence Research. — 2004. — Vol. 22. — P. 457–479. — Access mode: https://doi.org/10.1613/JAIR.1523.

[53] Heterogeneous Graph Neural Networks for Extractive Document Summarization / Wang Danqing, Liu Pengfei, Zheng Yining, Qiu Xipeng, and Huang Xuanjing // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics / ed. by Jurafsky Dan, Chai Joyce, Schluter Natalie, Tetreault Joel. — Online : Association for Computational Linguistics. — 2020. — July. — P. 6209–6219. — Access mode: https://aclanthology.org/2020.acl-main.553/.

[54] Liu Yang, Lapata Mirella. Hierarchical Transformers for Multi-Document Summarization // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics / ed. by Korhonen Anna, Traum David, Màrquez Lluís. — Florence, Italy : Association for Computational Linguistics. — 2019. — July. — P. 5070–5081. — Access mode: https://aclanthology.org/P19-1500/.

[55] From Local to Global: A GraphRAG Approach to Query-Focused Summarization / Edge Darren, Trinh Ha, Cheng Newman, Bradley Joshua, Chao Alex, Mody Apurva, Truitt Steven, Metropolitansky Dasha, Ness Robert Osazuwa, and Larson Jonathan // arXiv preprint arXiv:2404.16130. — 2024. — Access mode: https://arxiv.org/abs/2404.16130.

[56] Bratanič Tomaž, Nathan Paco. Essential GraphRAG: Knowledge Graph-Enhanced RAG. — First edition ed. — Shelter Island, NY : Manning Publications, 2025. — ISBN: 978-1-63343-626-8.

[57] Boosting Multi-Document Summarization with Hierarchical Graph Convolutional Networks / Song Yingjie, Yang Li, Luo Wenming, Xiao Xiong, and Tang Zhuo // Neurocomputing. — 2025. — Vol. 614. — P. 128753.

Приложение А. Вспомогательные работы и алгоритмы

Работа с форматом S2ORC и базой данных Semantic scholar. Датасеты Semantic Scholar Academic Graph, S2AG (https://www.semanticscholar.org/) созданы Allen AI Institute (https://allenai.org/) для обеспечения удобной работы с научными статьями. Представлены более 100 млн.статей с полным текстом и более 450 млн. статей с метаданными (полные графы цитирования, авторы, источники публикации, аннотации, векторные представления - эмбеддинги).

JSON-формат S2ORC (Semantic Scholar Open Research Corpus) выбран в настоящей работе для представления научных статей по следующим причинам:

- формат содержит четкую структуру метаданных, в том числе разделение статьи на разделы, абзацы и граф цитирования;
- возможность обогащения статьи, переведенной из формата PDF в S2ORC, за счет данных из базы Semantic scholar.

В рамках настоящей работы создан инструментарий по работе с форматом S2ORC[6]:

1. скрипт `convert_papers_to_s2orc.py` для конвертации текста статьи в формате PDF (с использованием библиотеки GROBID) или LaTeX в формат S2ORC-JSON;
2. скрипт `fetch_paper_data.py` - пайплайн для подготовки текста научной статьи для автоматизированной обработки: конвертация PDF с использованием `convert_papers_to_s2orc.py`, поиск по arXiv ID из имени и поиск по заголовку в arXiv с последующим запросом в базе данных Semantic Scholar по API, объединение данных в итоговый S2ORC-JSON;
3. скрипт `download_s2_datasets.py` для скачивания датасетов Semantic scholar на локальное хранилище.

Поскольку интеллект-карты (иерархические сводки) «золотого стандарта» подготавливались разметчикам в различных форматах и приложениях, был подготовлен конвертер[7] интеллект-карт форматов ‘.mmap‘,

[6]https://gitlab.mlsa-iai.ru/mindmaps/s2orc
[7]https://gitlab.mlsa-iai.ru/mindmaps/map2json-converter

‘.xmind‘, ‘.pdf‘, ‘.png‘, ‘.md‘, ‘.docx‘ в иерархическую сводку в формате JSON, совместимую с основным проектом. Решение данной задачи позволило сравнивать иерархические сводки, созданные алгоритмом, с «золотым стандартом».

Генерация иерархических сводок исключительно с помощью БЯМ. В ходе работы над проектом «Мастерская знаний» совместно с Ф.А. Соболевским, К.В. Воронцовым, М.С. Пукемо была проведена работа по созданию и настройке пайплайна генерации иерархических сводок с помощью БЯМ[8].

Был использован метод последовательного промптинга, в ходе которого модели SOTA (Claude Opus 4.6, OpenAI GPT 5.4) достигли результатов, сравнимых по метрикам со сводками из «золотого стандарта». Существенным минусом такого подхода является существенный расход токенов на подготовку одной сводки, что делает процедуру экономически нецелесообразной для регулярного использования.

Модели, не являющиеся SOTA (LlaMa 3.1-70B, Qwen 3.5, DeepSeek V3) не достигли приемлемого результата ни в сравнении с «золотым стандартом», ни с иерархическими сводками, сгенерированными алгоритмом из настоящей работы.

[8]https://gitlab.mlsa-iai.ru/mindmaps/LLM-single-doc-summary

Приложение Б. Подготовка иерархических сводок «золотого стандарта»

Основными свойствами иерархической сводки являются:

- сворачиваемость: иерархическую сводку можно произвольным образом свернуть с минимальными потерями содержательности, а при изучении сводки по любую глубину пользователь должен получить максимально полную сводку информации из документа соответствующего объема;
- неизбыточность: информация в иерархической сводке представляется в максимально кратком виде без потери содержания: формулировки в каждой из вершин должны быть объемлющими, краткими, информация в сводке не должна дублироваться;
- связность: иерархическая сводка, раскрытая произвольным образом, читается как связный текст при чтении в порядке обхода в глубину.

Иерархическую сводку можно строить от общего к частному, от частного к общему или комбинируя методы. Метод построения от общего к частному - выделить главную мысль и затем итеративно добавлять детали: сначала выделяется главная мысль документа, которая объявляется корневой вершиной, затем выделяются ключевые моменты, которые становятся вторым уровнем иерархии, затем информация в каждой вершине итеративно уточняется информацией из документа в дочерних вершинах. Метод построения от частного к общему идет от обратного, но результат должен получиться тот же - единая иерархия с общей корневой вершиной.

В ходе работы над проектом «Мастерская знаний» была проведена разметка (составление иерархических сводок) четырьмя разметчиками. Всего было размечено 20 научных статей, в основном по тематике многодокументной суммаризации. По 14 из статей было подготовлено по 2 и более иерархические сводки, которые вошли в «золотой стандарт».

Приложение В. Сравнение иерархических сводок, созданных различными алгоритмами и системами с «золотым стандартом»

Методика сравнения. Все иерархические сводки для сравнения создавались по статье Boosting multi-document summarization with hierarchical graph convolutional networks [57]. Статья представляет научное исследование, посвященное построению иерархических графов знаний в задаче много-документной суммаризации с использованием графовых сверточных нейронных сетей с учетом как связей между предложениями в одном документе, так и учетом связей между исследуемыми документами.

Для сравнения брались иерархические сводки, подготовленные экспертами (т.н. «золотой стандарт»). Стоит отметить, что сводки, созданные экспертами, хоть и созданы по единой методике, отраженной в Приложение Б. Подготовка иерархических сводок «золотого стандарта» не идентичны, поскольку каждый из экспертов может делать акцент на различные аспекты научной публикации, а также использует различные лексические конструкции при перефразировании предложений из исходного документа.

Использовались метрики TTED, STTED и BERTScore (основной набор), учитывающие структурное совпадение дерева иерархической сводки, и лексические метрики семейства ROUGE для установления совпадения отдельных элементов и уровней иерархии дерева.

Сравнение проводилось после генерации иерархической сводки в отдельном Jupyter Notebook в среде Google Colab по причине запуска модели sentence-transformer, требующей для работы GPU.

Если для одного документа задано несколько эталонных сводок, итоговая оценка строится по принципу «лучшего совпадения с эталоном». Для каждой метрики m и каждой глубины k вычисляется величина

$$m^*(C, \mathcal{G}; k) = \begin{cases} \min\limits_{j=1,\dots,M} m(C, G_j; k), & \text{если } m \text{ является дистанцией,} \\ \max\limits_{j=1,\dots,M} m(C, G_j; k), & \text{если } m \text{ является мерой сходства.} \end{cases}$$

Следовательно:

- для TTED, STTED — структурных метрик выбирается минимальное значение;
- для BERTScore, ROUGE — метрик лексической и семантической

схожести — максимальное.

Такой способ агрегации особенно уместен для рассматриваемой задачи, поскольку один и тот же документ может допускать несколько корректных вариантов иерархического разбиения и формулировки тем.

Кроме сравнения сгенерированной сводки с «золотым стандартом» дополнительно вычисляются попарные оценки между самими эталонными сводками. Для этого для всех пар

$$(G_i, G_j), \qquad i < j,$$

считаются те же самые метрики, после чего формируется усреднённая величина . Этот шаг особенно важен методологически: он позволяет интерпретировать качество автоматической системы не само по себе, а на фоне естественного межэкспертного разброса. Если кандидат показывает значения, близкие к , это означает, что система приближается к диапазону различий, наблюдаемому между разными экспертами-разметчиками.

Иерархическая сводка, созданная алгоритмом «minigraph» из настоящей работы в различных режимах создания локального графа. Сравниваются два построения иерархической сводки, в одном случае локальный граф создавался с помощью эвристического метода: `Эвристика`, в другом - с помощью `БЯМ`.

Таблица 1: Сводные значения метрик (среднее по уровням иерархии $= 2, \ldots, 8$). Стрелка ↓ — меньше-лучше, ↑ — больше лучше. Жирным шрифтом отмечен лучший результат.

Метрика	`Эвристика`	`БЯМ`	`Эксперты`	Δ лучшая − Эксперты
TTED ↓	0.682	0.683	**0.677**	+0.006
STTED ↓	0.731	0.762	**0.721**	+0.041
BERTScore ↑	0.867	**0.872**	0.862	+0.011
ROUGE-1 ↑	0.319	0.342	**0.404**	−0.063
ROUGE-2 ↑	0.093	0.109	**0.209**	−0.100
ROUGE-L ↑	0.286	0.319	**0.366**	−0.047

Значения метрик показывают, что подход к созданию локального графа с помощью `БЯМ` превосходит `Эвристику` по BERTScore и всем трём вариантам ROUGE, причём относительное преимущество по ROUGE-1/2/L достигает 7–17%. Это свидетельствует о том, что содержание сводки, где локальные графы извлекались БЯМ, лексически и семантически ближе к эталонной разметке.

Одновременно эвристический метод извлечения локального графа показывает немного меньшие значения структурных метрик TTED и STTED, то есть строит дерево сводки структурно построено ближе к «золотому стандарту».

Можно сказать, что разница между подходами к извлечению локальных графов из исходного документа несущественна и в случае необходимости экономии при работе приложения стоит выбирать эвристический метод, не затрачивая токены в случае с БЯМ.

Иерархическая сводка, созданная сервисом Mapify. В ходе работы проекта «Мастерская знаний» было проведено тестирование различных сервисов построения интеллект-карт и выбран лидер по качеству построения - Mapify[9].

С помощью сервиса была построена интеллект-карта, сохраненная на локальный диск и сконвертированная в формат JSON. Результат сравнивается с «золотым стандартом».

Таблица 2: Сводные значения метрик (среднее по уровням иерархии $= 2, \ldots, 8$). Стрелка $\downarrow$ — меньше-лучше, $\uparrow$ — больше лучше. Жирным шрифтом отмечен лучший результат.

Метрика	Mapify	`Эксперты`	Δ лучшая $-$ Эксперты
TTED $\downarrow$	0.775	**0.677**	$+0.098$
STTED $\downarrow$	0.827	**0.721**	$+0.106$
BERTScore $\uparrow$	0.855	**0.862**	-0.006
ROUGE-1 $\uparrow$	0.328	**0.405**	-0.077
ROUGE-2 $\uparrow$	0.110	**0.209**	-0.100
ROUGE-L $\uparrow$	0.270	**0.366**	-0.096

Иерархическая сводка, сгенерированная сервисом Mapify, уступает «золотому стандарту» сразу по всем шести метрикам. Отставание особенно велико по ROUGE-2 (-0.100) и ROUGE-L (-0.096), что свидетельствует о существенном лексическом расхождении с эталонной разметкой даже на уровне биграмм и общей логики изложения. По BERTScore разрыв минимален (-0.006): семантически сервис воспроизводит содержание статьи близко к тому, как это делает эксперт, но иными формулировками.

Структурные метрики TTED и STTED у Mapify заметно выше ($+0.098$ и $+0.106$ соответственно). Как видно, дерево Mapify содержит

[9]https://mapify.so

много узлов, не имеющих соответствия в экспертной разметке. Причина в данном случае - формальный подход сервиса к построению интеллект-карты: берется за основу логическое деление статьи на разделы, далее на подразделы, отдельные абзацы и все это переносится на древовидную структуру, которая становится более развернутой, мелкие пункты, дополнительная вложенность. Такой метод не позволяет решить задачу структурированного запоминания, не выделяет главные идеи статьи.

Приложение Г. Иерархические сводки в виде текстовых деревьев и интеллект-карт

Код приложения, разработанного в ходе исследования, размещен в репозитории на платформе GitHub[10]. В репозитории в папке `/summary` размещены иерархические сводки, рассмотренные выше при сравнении метрик, в форматах .json (текстовое дерево) и .pdf - визуализация в форме интеллект-карты. Имена файлов для каждой сводки приведены ниже:

- «Золотой стандарт» 1 - Сводка, созданная А.А. Костиным - 01_song25boosting_Kostin (.json / .pdf);
- «Золотой стандарт» 2 - Сводка, созданная Ф.А. Соболевским - 02_song25boosting_Sobolevsky;
- Сводка, созданная алгоритмом «minigraph» из настоящей работы и построением локальных графов с помощью БЯМ - minigraph_LLM_backend;
- Сводка, созданная алгоритмом «minigraph» из настоящей работы и построением локальных графов эвристическим методом - minigraph_heuristic_backend;
- Сводка, созданная сервисом Mapify - mapify.

[10]https://github.com/alexkostin78/minigraph