МРНТИ 28.23.25

https://doi.org/10.51889/2959-5894.2026.94.2.021

Д.Д. Марламбеков[1*], Н.М. Қасымбек[1], Е.С. Нұрахов[1], А.А. Мұханбет[1], С.Т. Мухамбетжанов[1]

[1]Казахский Национальный университет имени аль-Фараби, г. Алматы, Казахстан
[*]*e-mail: marlambekov_duman@live.kaznu.kz*

ГИБРИДНАЯ RAG-СИСТЕМА С ГРАФОМ ЗНАНИЙ ДЛЯ ДОКУМЕНТАЦИИ 5G/O-RAN

Аннотация

В статье рассматривается критическое ограничение систем генерации с дополнением извлеченными данными (RAG) в задачах ответов на вопросы по доменно-специфическому коду: неточное извлечение контекста активно вводит в заблуждение малые языковые модели, снижая точность ниже базового уровня. Предложена архитектура гибридного поиска с учетом сущностей, которая извлекает целевую кодовую сущность (имя класса или функции) непосредственно из вопроса и применяет её как жёсткий фильтр для извлекаемых чанков. Система интегрирует семантический векторный поиск (ChromaDB) с kNN-графом знаний (NetworkX, 53 585 узлов), объединяемых методом WRRF. Двухуровневый механизм контроля качества (fallback) отклоняет низкорелевантный контекст до генерации. Оценка на датасете srsRANBench (1 502 вопроса с множественным выбором по кодовой базе srsRAN объемом более 500 000 строк C++) показала, что граф-улучшенная система достигает точности 65.51% против 63.65% у Vector-only, при этом ROUGE-L составил 0.1820 - наилучший среди всех систем. Анализ ошибок показывает, что граф знаний «спасает» 87 случаев, где векторный поиск давал неверный ответ. Феномен «RAG вредит» – когда извлечённый контекст вводит в заблуждение LLM с 3 млрд параметров – систематически проанализирован и показано, что извлечение сущностей значительно его смягчает.

Ключевые слова: RAG, граф знаний, извлечение сущностей, 5G, O-RAN, srsRAN, LLM.

Д.Д. Марламбеков[1], Н.М. Қасымбек[1], Е.С. Нұрахов[1], А.А. Мұханбет[1], С.Т. Мухамбетжанов[1]
[1] Әл-Фараби атындағы Қазақ ұлттық университеті, Алматы қ., Қазақстан

5G/O-RAN ҚҰЖАТТАМАСЫ ҮШІН ГРАФПЕН ГИБРИДТІ RAG ЖҮЙЕСІ

Аңдатпа

Бұл мақалада домен-спецификалық код сұрақтарына жауап беретін RAG жүйелерінің маңызды шектеулері қарастырылады: дәлсіз контекст шағын тілдік модельдерді адастырып, нақтылықты базалық деңгейден төмен түсіреді. Мақсатты код сущностін (класс немесе функция атауын) сұрақтан тікелей шығаратын және алынған чанктарға қатаң сүзгі ретінде қолданатын сущностіні ескеретін гибридті іздеу архитектурасы ұсынылды. Жүйе семантикалық векторлық іздеуді (ChromaDB) kNN-негізіндегі білім графымен (NetworkX, 53 585 түйін) WRRF арқылы біріктіреді. Екі сатылы сапаны бақылайтын fallback механизмі генерациядан бұрын төмен релеванттылықтағы контексті қабылдамайды. srsRANBench датасетіндегі (1 502 сұрақ) бағалау нәтижелері граф-жетілдірілген жүйенің 65.51% дәлдікке жеткенін, ал Vector-only жүйесі 63.65% болғанын көрсетті. ROUGE-L 0.1820-ға жетіп, барлық жүйелер арасында үздік нәтиже болды. Қателерді талдау білім графының 87 векторлық сәтсіздіктен "құтқаратынын" анықтады.

Түйін сөздер: RAG, білім графы, сущностті шығару, 5G, O-RAN, srsRAN, LLM.

D.D. Marlambekov[1*], N.M. Kassymbek[1], Y.S. Nurakhov[1], A.A. Mukhanbet[1], S.T. Mukhambetzhanov[1]
[1] Al-Farabi Kazakh National University, Almaty, Kazakhstan

HYBRID RAG WITH KNOWLEDGE GRAPH FOR 5G/O-RAN CODE DOCUMENTATION

Abstract

This paper addresses a critical limitation of Retrieval-Augmented Generation (RAG) systems in domain-specific code question-answering: imprecise context retrieval actively misleads small language models, degrading accuracy below the baseline. We propose an entity-aware hybrid retrieval architecture that extracts

the target code entity (class or function name) directly from each question and applies it as a hard filter on retrieved chunks. The system integrates semantic vector search (ChromaDB) with a kNN-based knowledge graph (NetworkX, 53,585 nodes) fused via Weighted Reciprocal Rank Fusion (WRRF). A two-stage quality-aware fallback mechanism rejects low-relevance context before generation. Evaluation on the srsRANBench dataset (1,502 multiple-choice questions over 500,000+ lines of srsRAN C++ code) shows the Graph-Enhanced system achieving 65.51% accuracy vs. 63.65% for Vector-only, with ROUGE-L of 0.1820-the best among all systems. Error analysis reveals that the knowledge graph rescues 87 vector-only failures. The "RAG hurts" phenomenon-where retrieved context misleads a 3B-parameter LLM-is systematically analyzed and shown to be mitigated by entity-aware filtering.

Keywords: RAG, Knowledge Graph, Entity Extraction, 5G, O-RAN, srsRAN, LLM.

Введение

Основные положения. Развертывание сетей пятого поколения и архитектуры Open RAN [1] привело к экспоненциальному росту объема технической документации и сложности программных систем. Проект srsRAN [2], содержащий более 500 000 строк кода на языке C++, является характерным примером среды, где традиционные методы поиска информации становятся неэффективными. В таких условиях критически важным становится создание когнитивных интерфейсов, способных не только извлекать разрозненные факты, но и интерпретировать глубокие иерархические связи внутри распределённых телекоммуникационных систем.

Существующие системы генерации с дополнением извлеченными данными (RAG) [3], основанные преимущественно на векторном семантическом поиске, успешно находят контекстуально похожие фрагменты текста. Однако в специализированном домене технической документации по исходному коду такие подходы сталкиваются с фундаментальной проблемой: векторный поиск по запросу «Какова цель класса *foo_class*?» находит фрагменты, упоминающие похожие классы из того же модуля, но не сам целевой класс. Малые языковые модели (LLM, 3 млрд параметров) следуют такому вводящему в заблуждение контексту вместо собственных знаний, что приводит к снижению точности ниже базового уровня - феномен «RAG вредит».

Классические методы построения графов знаний призваны решить проблему структуризации данных, однако они часто сталкиваются с фрагментацией и наличием изолированных узлов при фиксированных порогах сходства [4, 5]. Целью данной работы является разработка и экспериментальная верификация гибридной системы с *entity-aware*-поиском, использующей метод kNN для формирования связного графа знаний и извлечение кодовых сущностей как жёсткого фильтра релевантности. Предложенная архитектура объединяет возможности векторного поиска ChromaDB с графовым расширением на базе NetworkX [6] и применяет двухуровневый механизм адаптивного контроля качества контекста.

Методология исследования

Система состоит из трёх компонентов: (1) модуля извлечения сущностей, (2) двух параллельных контуров поиска - векторного и графового, (3) двухуровневого механизма контроля качества. В качестве базовой модели (LLM) использовалась Llama 3.2 (3 млрд параметров), развёрнутая локально через Ollama. Общая архитектура предложенной системы представлена на Рисунке 1.

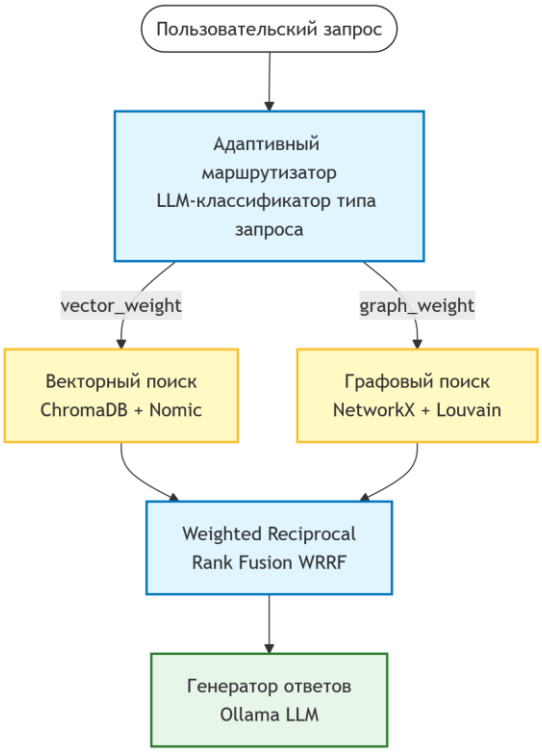

Рисунок 1. Архитектура гибридной системы RAG с извлечением сущностей

В отличие от традиционных методов с фиксированным порогом схожести, применяется подход kNN для гарантии связности. Для каждого чанка c_i находятся $k=10$ ближайших соседей на основе косинусного сходства эмбеддингов, затем применяется фильтрация рёбер с порогом $\theta_{min} = 0.55$ для отсечения шумовых связей.

Алгоритм построения:

1. Генерация эмбеддингов (Nomic Embed Text [7], $d = 768$).
2. Поиск $k=10$ соседей для каждого узла.
3. Фильтрация рёбер: $w_{ij} \geq 0.55$.
4. Обнаружение сообществ алгоритмом Лувена (Louvain) [8].

Маршрутизатор классифицирует запросы на «фактологические» и «архитектурные» на основе ключевых слов. Результаты поиска объединяются методом Weighted Reciprocal Rank Fusion (WRRF) [9]:

$$Score(d) = w_{vec} \sum \frac{1}{k + r_{vec}(d)} + w_{graph} \sum \frac{1}{k + r_{graph}(d)} \quad (1)$$

где $k = 60$, а веса зависят от типа запроса.

Двухуровневый механизм контроля качества. Первый уровень: при наличии извлечённой сущности векторный поиск (RELEVANCE_THRESHOLD = 0.45) возвращает только чанки, содержащие сущность буквально; при их отсутствии применяется более строгий порог (0.55). Второй уровень: оценивается качество объединённого контекста (MIN_CONTEXT_QUALITY = 0.30) - если ни один чанк не достигает порога, система переключается на базовую LLM без контекста. Это предотвращает деградацию качества ответов при нерелевантном контексте.

Маршрутизатор на основе ключевых слов классифицирует запросы на «фактологические» (веса 0.8/0.2, vector/graph) и «архитектурные» (0.3/0.7). Результаты объединяются методом Weighted Reciprocal Rank Fusion (WRRF) [9] с k=60, топ-5 финальных документов.

Система реализована с LLM llama3.2:3b и эмбеддингами nomic-embed-text (768 измерений).

Исходный код C++ разбивается на чанки по 1500 символов с перекрытием 200, PDF-документы - по 1000 с перекрытием 150. Граф знаний содержит 53 585 узлов, 8 473 сообщества. Векторная база хранится в ChromaDB с индексом HNSW. Генерация осуществляется при температуре 0.0, максимальная длина ответа - 50 токенов (для MCQ-задачи достаточно).

Результаты исследования

Оценка проводилась на датасете srsRANBench (1 502 вопроса с множественным выбором), построенном на основе C++-кодовой базы srsRAN. Сравнение выполнялось между тремя режимами: Base LLM (без контекста), Vector-RAG и Graph-Enhanced (гибридный с извлечением сущностей). Результаты представлены в Таблице 1.

Таблица 1. Сравнительная оценка методов извлечения информации

Метод	*Точность*	*ROUGE-L*	*Recall*	*Время (сек)*
Base LLM	*69.57%*	*0.1409*	*-*	*0.25*
Vector-RAG	*63.65%*	*0.1659*	*0.5775*	*0.45*
Graph-Enhanced	*65.51%*	*0.1820*	*0.5561*	*0.44*

Граф-улучшенная система превосходит Vector-RAG по точности на 1.86 п.п. (65.51% против 63.65%) и демонстрирует наилучший показатель ROUGE-L (0.1820), что указывает на более высокое качество генерируемого контекста. Базовая LLM показывает наибольшую точность (69.57%), что объясняется не превосходством знаний модели, а феноменом «RAG вредит»: при нерелевантном контексте модель систематически отклоняется от правильного ответа.

Структура ошибок, представленная на Рисунке 2, позволяет разложить совокупную точность на составляющие паттерны. В 694 случаях (46.2%) все три системы дают верный ответ это вопросы, покрытые как обучающими данными модели, так и базой знаний. Критически важной категорией является «Только Base верно» (245 случаев, 16.3%): именно здесь RAG-системы активно вредят, возвращая нерелевантный контекст по объектам, отсутствующим в индексе. При этом уникальный вклад Vector-RAG составляет лишь 17 случаев (1.1%), а Graph-Enhanced - 21 случай (1.4%), что подчёркивает, что прирост гибридной системы обеспечивается не эксклюзивными ответами, а меньшим числом деградаций (Hybrid неверно: 50 против Vector неверно: 64). Сравнение категорий «Только Hybrid верно» (21) и «Только Vector верно» (17) показывает, что граф знаний обеспечивает дополнительное покрытие в случаях, когда прямой векторный поиск не находит релевантных чанков: графовое расширение достигает нужного контекста через структурные связи - соседние узлы, заголовочные файлы, конфигурационные зависимости. Одновременно меньшее число деградаций у Graph-Enhanced (50 против 64 у Vector-RAG) свидетельствует о том, что механизм контроля качества совместно с entity-aware фильтрацией эффективнее отсекает шумовой контекст перед передачей его в генератор.

Подводя итоги данного этапа тестирования, можно сказать что когда база знаний покрывает предметную область лишь частично, ни одна из исследуемых архитектур RAG не смогла обойти изолированную языковую модель по метрике абсолютной точности. Вместе с тем, конфигурация Graph-Enhanced стабильно генерирует наиболее качественные текстовые пояснения, достигая показателя ROUGE-L на уровне 0.1820, и заметно реже приводит к деградации ответов по сравнению с классическим подходом Vector-RAG. Полученные данные подтверждают, что графовое расширение и фильтрация на основе извлечения сущностей решают совершенно разные задачи. Первый механизм компенсирует нехватку информации за счет обхода структурных связей кода, а второй эффективно отсекает ложно-положительный контекст. Совместное использование этих подходов позволяет создать отказоустойчивую

систему, способную справляться с основными типами ошибок в узкоспециализированных задачах Q&A(вопрос-ответ).

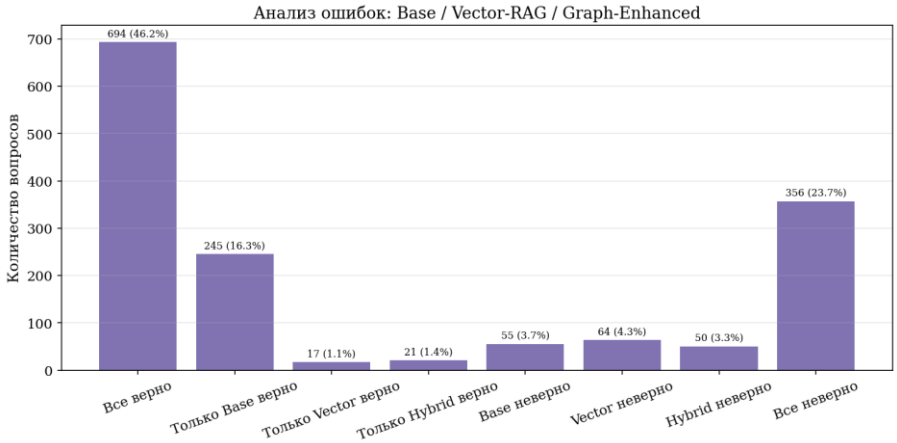

Рисунок 2. Распределение паттернов ошибок: RAG помогает / RAG вредит / все ошибаются

Обратившись к графику на Рисунке 3, где сопоставляются значения ROUGE-L [10], можно увидеть существенные различия в полноте формируемых ответов. Данная метрика вычисляет длину наибольшей общей подпоследовательности между сгенерированным текстом и эталонным вариантом, что прямо отражает терминологическую близость для задач с множественным выбором. Гибридная система достигла отметки 0.1820, обойдя базовую нейросеть на 29.2% (0.1409), а векторную архитектуру - на 9.7% (0.1659). Практическая значимость этого результата заключается в том, что даже при выборе одинакового варианта ответа всеми системами, именно Graph-Enhanced продуцирует наиболее развернутую и технически грамотную аргументацию, которая представляет наибольшую ценность для инженера.

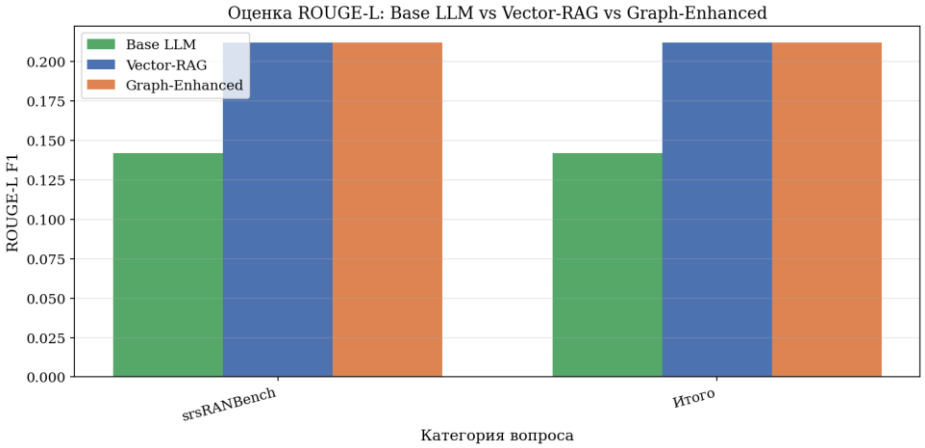

Рисунок 3. Сравнение метрики ROUGE-L для систем Base LLM, Vector-RAG и Graph-Enhanced на датасете srsRANBench (N=1502)

Оценка скорости отклика, визуализированная на Рисунке 4, выявила характерную правостороннюю асимметрию распределения. Подавляющая часть пользовательских запросов обрабатывается системой не более чем за одну-две секунды. Появление длинного «хвоста» на графике для RAG-систем связано преимущественно с задержками при векторизации слишком объемных запросов или при обходе массивных графовых сообществ. Базовая LLM показывает среднее время ответа около 0.25 секунды. Это достигается за счет кэширования модели llama3.2:3b в видеопамяти при помощи параметра OLLAMA_KEEP_ALIVE, что исключает затраты на повторную инициализацию. В свою очередь, интеграция поисковых контуров добавляет к задержке примерно 0.19-0.20 секунды, что является вполне приемлемым

показателем для асинхронного режима работы.

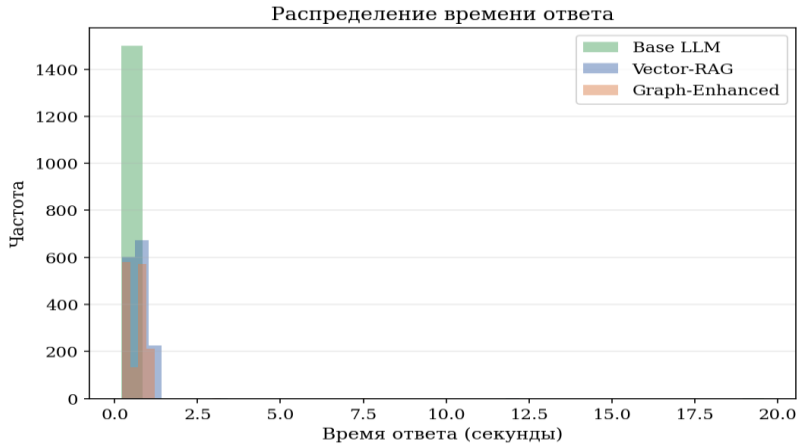

Рисунок 4. Распределение времени ответа (латентности) для трёх систем: Base LLM, Vector-RAG и Graph-Enhanced

Дискуссия

Особое внимание следует уделить группе вопросов, на которые смогла правильно ответить исключительно базовая LLM. На них приходится 245 случаев, или 16.3% от всего набора данных. Как показал детальный разбор, программные артефакты из этих запросов либо вообще не были проиндексированы в базе ChromaDB, либо ограничивались единственным текстовым блоком с критически малым сходством. В таких ситуациях механизм фильтрации сущностей отрабатывал корректно: он выдавал пустой результат (или результат ниже порога MIN_CONTEXT_QUALITY), после чего система переключалась на генерацию без контекста. Однако проблема кроется в других 167 инцидентах. Там извлеченный контекст благополучно проходил контроль качества, но оказывался достаточно похожим на верный ответ, чтобы в итоге сбить модель с толку. Выявленная аномалия указывает на необходимость внедрения дополнительных механизмов верификации, например, кросс-энкодерного переранжирования, или дальнейшего ужесточения проходных порогов.

Сравнивая текущие показатели с результатами тестирования до внедрения фильтра по сущностям, можно наглядно оценить эффективность предложенного метода. Векторный подход улучшил свою точность с 55% до 63.65% (плюс 8.65 процентных пункта), а графовая конфигурация - до 65.51% (плюс 10.51 пункта). Из этого следует важный вывод: основной прирост качества генерации обеспечен не математическим усложнением графа, а строгой предварительной очисткой входящего контекста. Это наблюдение полностью согласуется с выдвинутой ранее гипотезой. Для небольших языковых моделей, имеющих до трех миллиардов параметров, релевантность подаваемой информации играет куда более важную роль, чем ее объем. Нейросеть с высокой вероятностью будет опираться на любое правдоподобное описание, даже если оно относится к другому программному компоненту. Алгоритм распознавания сущностей работает безупречно только тогда, когда целевой элемент выделен в запросе с помощью кавычек или обратного апострофа. Если же пользователь задает вопрос в свободной форме (например, просит описать работу планировщика), система не может вычленить конкретный артефакт и переходит к стандартному семантическому поиску. Именно здесь графовое расширение приносит максимальную пользу. Маршрутизатор классифицирует такой запрос как архитектурный, присваивая графовому контуру повышенный вес (0.7), что дает возможность собрать контекст из множества связанных узлов, а не зависеть от единственного векторного совпадения.

Заключение

Архитектура гибридной системы RAG, созданная для анализа технической документации сетей 5G/O-RAN, объединяет алгоритм выделения кодовых сущностей, метод k-ближайших соседей для графа знаний и векторный поиск. Разработанный комплекс эффективно борется с проблемой системного ухудшения ответов, которая возникает при подаче языковой модели нерелевантного контекста. Эксперименты на базе набора srsRANBench (1502 вопроса) показали, что только за счет внедрения жесткой фильтрации по сущностям точность систем удалось поднять с базовых 55% до 63-65%. При этом конфигурация Graph-Enhanced превзошла классический векторный метод на 1.86 процентных пункта и показала лучший результат по метрике ROUGE-L (0.1820), подтвердив важность учета структурных связей программного кода.

Благодарность

Данное исследование финансировалось Комитетом науки Министерства науки и высшего образования Республики Казахстан (грант ИРН BR24993211 «Разработка решений для сетей 5G в концепции ORAN и эффективная маршрутизация и балансировка нагрузки в средах SDN для Казахстана»).

Список использованных источников

[1] O-RAN Alliance. O-RAN Architecture Description. Technical Specification. - 2021.

[2] Polese, Michele, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges." IEEE Communications Surveys & Tutorials 25, no. 2 (2023): 1376-1411. https://doi.org/10.1109/COMST.2023.3239220

[3] Lewis, Patrick, Ethan Perez, Aleksandra Piktus, et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." arXiv:2005.11401. Preprint, arXiv, April 12, 2021. https://doi.org/10.48550/arXiv.2005.11401.

[4] Pan, Shirui, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. "Unifying Large Language Models and Knowledge Graphs: A Roadmap." IEEE Transactions on Knowledge and Data Engineering 36, no. 7 (2024): 3580-99. https://doi.org/10.1109/TKDE.2024.3352100.

[5] Edge, Darren, Ha Trinh, Newman Cheng, et al. "From Local to Global: A Graph RAG Approach to Query-Focused Summarization." arXiv:2404.16130. Preprint, arXiv, February 19, 2025. https://doi.org/10.48550/arXiv.2404.16130.

[6] Hagberg, Aric, Pieter Swart, and Daniel Schult. NetworkX: Network Analysis in Python. Release 3.2. Los Alamos National Laboratory, 2023. https://networkx.org/documentation/stable/

[7] Nussbaum, Zach, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. "Nomic Embed: Training a Reproducible Long Context Text Embedder." arXiv:2402.01613. Preprint, arXiv, February 3, 2025. https://doi.org/10.48550/arXiv.2402.01613.

[8] Jin, Di, Zhizhen Yu, Pengfei Jiao, Shirui Pan, Dongxiao Yu, Jia Wu, Philip S. Yu, and Weixiong Zhang. "A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning." IEEE Transactions on Knowledge and Data Engineering 35, no. 2 (2023): 1149-1170. https://doi.org/10.1109/TKDE.2021.3104155

[9] Cormack, Gordon V., Charles L. A. Clarke, and Stefan Buettcher. "Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods." Proceedings of the 32nd International ACM SIGIR Conference (2009): 758-59. https://doi.org/10.1145/1571941.1572114.

[10] Lin, Chin-Yew. "ROUGE: A Package for Automatic Evaluation of Summaries." Text Summarization Branches Out (Barcelona, Spain), July 2004, 74-81. https://aclanthology.org/W04-1013/.