ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет Санкт-Петербургская школа физико-математических

и компьютерных наук

Кощенко Екатерина Васильевна

ИСПОЛЬЗОВАНИЕ СОЧЕТАНИЯ РАЗЛИЧНЫХ ТИПОВ ДАННЫХ ДЛЯ РЕКОМЕНДАЦИИ КАНАЛОВ

Выпускная квалификационная работа

по направлению подготовки 01.04.02 «Прикладная математика и информатика»

образовательная программа «Машинное обучение и анализ данных»

Рецензент

Ершов Василий Алексеевич

Научный руководитель

к.ф.-м.н., доц

Москвин Денис Николаевич

Консультант

Коваленко Владимир Владимирович

Санкт-Петербург 2021

Аннотация

Коллаборационные платформы, такие как GitHub и Slack, являются важными инструментами в работе сотрудников IT компаний. Данные, которые они агрегируют, могут быть использованы для построения различных алгоритмов социо-технической поддержки пользователей, упрощающих процесс работы сотрудников компаний. Однако, распределенность этих данных по разным платформам приводит к тому, что их объединения является трудоёмким процессом, в связи с чем существующие алгоритмы социо-технической поддержки (такие как рекомендация каналов в мессенджерах) основываются лишь на данных, напрямую связанных с целью алгоритмов.

В данной работе был собран датасет, содержащий в себе данные о взаимодействиях сотрудников компании JetBrains в каналах и репозиториях платформы Space, репозиториях GitHub, а также позиции пользователей в компании, полученные с помощью Space. На основе полученного датасета были построены несколько видов систем рекомендации каналов в Space, основанных на разных комбинациях модальностей. Лучшие из построенных моделей были объединены в одну мультимодальную систему рекомендации. Хотя оценка работы этой мультимодальной системы на исторических данных показала худшие метрики по сравнению с лучшим из унимодальных алгоритмов (матричная факторизация на каналах), пользователи Space оценили рекомендованные мультимодальной системой каналы значительно лучше, чем рекомендации матричной факторизации.

По результатам работы был сделан вывод, что использование данных о структуре организации и технических репозиториях пользователей позволяет бороться с переобучением системы рекомендации, а также решает проблему холодного старта пользователей.

Ключевые слова: алгоритмы социо-технической поддержки, система рекомендации каналов, мультимодальные данные.

Abstract

Nowadays collaborative platforms, such as GitHub and Slack, are a vital instrument in a day-to-day routine of every IT employee. As a result, data that is aggregated by these platforms has a significant value for algorithms of a socio-technical assistance that improve employees' working conditions. However, the distribution of this data across different platforms leads to the fact that combining it is a very time-consuming process, and therefore the existing algorithms for socio-technical assistance (such as recommendation systems for channels in messengers) are based only on data directly related to the purpose of the algorithms.

In this work, I collected a dataset containing data on interactions between JetBrains employees in channels and repositories on the Space platform, repositories on GitHub, as well as the positions of users in the company, obtained from Space. Several types of Space channel recommendation systems, based on different combinations of data modalities, were built using the collected dataset. The best of the constructed models were combined into a resulting multimodal channel recommendation system. Even though the evaluation of the performance of this multimodal system on historical data showed the worse metrics compared to the best of the unimodal algorithms (matrix factorization on channels), Space users rated the channels recommended by the multimodal system significantly better than the once suggested by matrix factorization.

Based on the results of this work, it was concluded that using the data on the organization structure and employees' technical repositories allows to mitigate the overfitting problem, and solves the problem of users' cold start.

Key words: socio-technical assistance algorithms, channel recommendation system, multimodal data.

ОГЛАВЛЕНИЕ

Введение

Постановка и актуальность проблемы

Сегодня сотрудники больших IT компаний на ежедневной основе используют различные коллаборационные платформы (мессенджеры, гит хостинги, трекеры задач и так далее). В результате эти платформы агрегируют большое количество информации, описывающей взаимодействие пользователей друг с другом и различными артефактами платформ. Комбинирование таких данных может дать хорошее представление социальных и технических связей внутри организаций. Применив это представление к алгоритмам, поддерживающим коллаборативную работу в компании, мы можем улучшить их качество. Данные алгоритмы будем называть ***алгоритмам социо-технической поддержки***. Яркими экземплярами таких алгоритмов являются разнообразные рекомендательные системы [1]. Например, системы рекомендации рецензентов (или, аналогично, экспертов) [2]. Рецензенты помогают следить за качеством кода программистов и корректировать их недочеты и ошибки. Очевидно, что правильно подобранные рецензенты (знающие область и, зачастую, знакомые с проектом) напрямую влияют на качество кода и развитие человека. Более того, чем больше сотрудников вовлечено в любой проект, тем меньше связанные с ним риски. Если над каким-то проектом (целиком или существенной его частью) работает лишь один человек, то его болезнь/увольнение/другая чрезвычайная ситуация может привести к замедлению или полной остановке проекта. Для описания таких ситуация придуман автобусный фактор (bus factor) [3]. Автобусный фактор — это мера сосредоточения информации среди отдельных членов проекта. Он показывает, сколько человек должно по какой-либо причине “исчезнуть”, чтобы проект был остановлен из-за недостатка компетентных сотрудников, информированных о проекте. Одним из эффектов алгоритмов социо-технической поддержки часто

является увеличение автобусного фактора, то есть уменьшение количества человек, работа которых критична для успешного завершения проекта.

Другим примером алгоритмов социо-технической поддержки являются ***системы рекомендации каналов*** в текстовых мессенджерах. Канал — это групповой чат, где все общение сфокусировано на конкретных проектах, темах или командах. Сообщения в каналах, для удобства поиска нужного обсуждения, обычно сгруппированы по веткам (тредам). Самым известным корпоративным мессенджером, основанным на системе каналов, является Slack [4]. Каналы удобны тем, что они избавляют пользователей от информационного шума и помогают сконцентрировать общение на конкретные темы в одном месте. Такая система кажется очень удобной, но у неё есть существенный недостаток: пользователи оказываются ограничены известными им каналами, они “не выглядывают” за границы своего пространства. Автоматические системы рекомендации каналов (без запроса пользователя) решают эту проблему. Релевантные и разнообразные рекомендации увеличивают осведомленность сотрудников о проектах друг друга и стимулируют взаимодействие между командами. А это, в свою очередь, может увеличить автобусный фактор. Если за проектом уволившегося сотрудника следили несколько человек из его команды (и получали регулярные обновления о состоянии работы), то они смогут продолжить этот проект с минимальными затруднениями. Другая ситуация: команда А начинает новый проект, а команда Б несколько месяцев назад закончила схожий проект. В идеальном мире, система рекомендации каналов должна отследить эту ситуацию и предложить пользователям из команды А каналы, где команда Б обсуждала их схожий проект.

Итак, качественные системы рекомендации каналов существенно улучшают взаимодействие сотрудников, что может привести к ускорению и повышению качества их работы и уменьшению рисков. И кажется очевидным, что чем больше информацию имеет система о сотрудниках, их работе и ежедневных профессиональных и социальных взаимодействиях, тем лучшего качества

рекомендации она сможет предоставить. Проблема состоит в том, что все эти социальные и технические данные *распределены по различным платформам*, поэтому обычно их довольно тяжело получить. Например, социальные взаимодействия сотрудников происходят в мессенджерах (Slack [4], Telegram [5]), работа и проекты хранятся на гит-хостингах (GitHub [6]), расписание и встречи поддерживаются в календарях (Google Calendar [7]), а для получения информации о структуре организации и позициях сотрудников в ней придётся воспользоваться официальным сайтом компании. Таким образом, с учётом взаимных интеграций, с увеличением числа используемых платформ, трудоёмкость сбора данных растёт квадратично. Поэтому большинство исследований в области алгоритмов социо-технической поддержки сконцентрированы только на одном типе данных [8]. Например, для рекомендации рецензентов кода обычно используются только информация, полученная из репозиториев [2] (список авторов, коммиты, код-ревью и т.д.). Однако, данные о социальных взаимодействиях сотрудников могли бы спровоцировать рекомендации рецензентов, имеющих экспертизу в близкой области из других проектов, но никогда не работавших над данным проектом. Аналогично, мессенджеры со встроенной системой рекомендации каналов используют только уже имеющиеся у них данные о самих каналах и не выходят за пределы их досягаемости. Более конкретно, Slack в своей статье [9] показывает, что при построении рекомендаций используются только знания о времени, проведенном пользователем в канале (для чтения и общения). Более подробно описанный Slack алгоритм разобран в Литературном обзоре разделе 2.1 (Рекомендательные системы). Потенциальным способом улучшения рекомендаций, основанных только на данных об общении в каналах, является использование структуры организации. Возможно, системы, учитывающие иерархию компании (например, позиции сотрудников), будут выдавать более релевантные рекомендации.

В последнее время, начинается тенденция объединения различных функциональностей в одну платформу. Примером того является Space [10]. ***Space***

— это интегрированная среда для командной работы от JetBrains. Она совмещает в себе много функциональностей различных типов, таких как гит хостинг, код ревью, таск трекер, календарь, каналы для общения, блоги, профили команд и так далее. Это значит, что, в перспективе, из Space можно будет извлечь самые разнообразные данные, а их совместная обработка будет не так трудоемка как раньше (при использовании множества платформ).

Цели и задачи

Целью данной работы является построить систему рекомендации каналов для платформы Space, работающую с данными трёх типов: каналы (их подписчики и сообщения, сгруппированные по тредам), технические репозитории (авторы, коммиты и исходный код) и структуры организации (позиции, команды и менеджеры сотрудников). Если система рекомендации каналов на мультимодальных данных трёх типов покажет лучшие результаты по сравнению с моделями, основанными на данных одного типа (или поможет с проблемой разнообразия рекомендаций), то эта идея может быть применена к другим алгоритмам социо-технической поддержки.

Для решения поставленной цели были изучены работы по обработке данных (глава 2). После этого были собраны данные для построения алгоритма (глава 3), на основе которых были построены несколько унимодальных систем рекомендации каналов (основанная на пользователях коллаборативная фильтрация, матричная факторизация, Node2Vec эмбеддинг, а также описанный Slack алгоритм), позже объединенных в одну мультимодальную модель (глава 4). Итоговая система была обучена на разных комбинациях данных (только каналы, каналы и репозитории, каналы и структура, и все вместе) и сравнена с унимодальными алгоритмами. В конце работы был сделан анализ информативности факторов, а также проведена оценка полученной рекомендательной системы пользователями Space.

2. Литературный обзор

2.1. Рекомендательные системы

Подход Slack.

Алгоритм рекомендации каналов в мессенджерах, описание которого опубликовал Slack в 2016 году [9], является вариацией коллаборативной фильтрации, основанной на сущностях (item-based collaborative filtering) [11]. На рисунке 2.1 представлена общая схема его работы. Его идея основана на применении kNN регрессии для заполнения отсутствующих значений в матрице релевантности пользователей и каналов (которые в поставленной задаче выполняют роль сущностей). В начале работы строится матрица, кодирующая соответствии пользователей и каналов. В качестве весов в матрице, авторы используют отношение времени, проведенного за написанием в канал, ко времени чтения. В Space такой информации так просто не получить, поэтому данное отношение было заменено на факторы, аналогично описывающие активность пользователя в канале (раздел 4.1). Примером такого фактора является отношение количества тредов с участием пользователя ко всем тредам, появившимся в канале после определенной даты. После заполнения матрицы взаимодействий, Slack алгоритм строит матрицу схожести каналов, используя для этого косинусную смежность векторов каналов, составленных из их оценок схожести со всеми пользователями (соответствующая строка описанной выше матрицы). Далее для оценки каждого канала будет использовано только k каналов, ближайших к целевому. Последним этапом модель заполняет пропуски в исходной матрице релевантностей, взвешивая схожести каналов, ближайших к целевому, на их известную изначально соотносимость с пользователем.

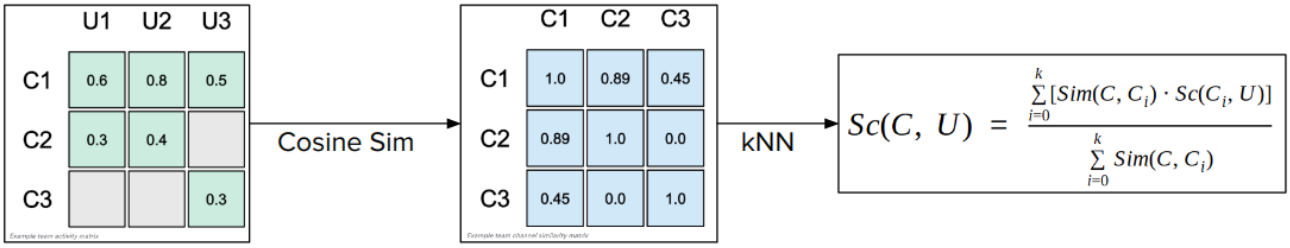

Рисунок 2.1. Схема работы Slack алгоритма.

Популярные подходы к построению рекомендательных систем.

Задача построения различных рекомендательных систем является всё более и более актуальной во всех областях, начиная от развлекательной (рекомендация фильмов [35]), заканчивая медицинской [36]. Каждый год появляются новые сложные нейронные модели рекомендации [37, 38], однако часто они дают результаты не существенно лучшие, чем более простые модели, в то время как на практике встраивание их в систему является очень трудоёмким процессом [39].

Популярным способом начала работы над построением рекомендаций в любой области является коллаборативная фильтрация двух типов [40]. Основанная на пользователях коллаборативная фильтрация подбирает для каждого пользователя близких ему (по оценкам рекомендуемых сущностей), и в качестве рекомендации предлагает популярные среди них объекты. Коллаборативная фильтрация, основанная на сущностях, подходит к задаче с другой стороны. Такие алгоритмы подбирают для каждого объекта набор схожих ему по каким-то их параметрам (или оценкам пользователей), а затем для набора уже оцененных каждым пользователем сущностей ищет другие, близкие ко всему этому набору.

Одним из способов коллаборативной фильтрации является матричная факторизация [30, 31, 32]. При построении рекомендаций у нас есть таблица релевантности сущностей для пользователей, заполненная данными, явно или неявно полученными из взаимодействия пользователей с системой. Проблему рекомендации можно переформулировать как задачу заполнения нулевых элементов этой матрицы оценками релевантности. Эта задача решается матричной

факторизацией, в результате которой получаются из матрицы релевантностей получаются две другие: векторные латентные представления пользователей и сущностей. После этого получить оценку релевантности любой сущности для любого пользователя можно, посчитав близость их векторных представлений.

2.2. Обработка данных

Репозитории данных о разработке, такие как системы контроля версий, баг-трекеры, системы код-ревью и системы непрерывной интеграции, служат источниками данных для множества подходов, призванных улучшить существующие инструменты разработки программного обеспечения. Методами сбора и анализа этих данных, а также их применением в практических контекстах, занимается большое сообщество исследователей. Кроме того, подобным методам посвящено несколько крупных международных конференций [12, 13, 14].

Помимо общих методов обработки данных, не являющихся специфическими для данных из программных репозиториев, таких как препроцессинг текста [15] и анонимизация данных [16], существуют специальные методы обработки данных из программных репозиториев. Среди примеров таких методов можно привести дедупликацию пользовательских сущностей в системах контроля версий [17, 18], подсчёт метрик кода и активности репозиториев [19], а также генерация производных представлений кода [20] для перевода их в формат, подходящий для дальнейшего анализа и использования в обучении моделей машинного обучения. Несмотря на высокую степень автоматизации инструментов разработки и относительную простоту извлечения данных из многих из них [21], эффективное извлечение большого количества точных исторических данных, таких как история версий артефактов [22], может представлять собой существенную техническую работу [23]. При анализе исторических записей о взаимодействии людей в процессе разработки, в зависимости от источника данных, также бывает необходима очистка и фильтрация [24].

Вне зависимости от предметной области и происхождения данных, для оптимальной производительности алгоритмов рекомендательных систем часто необходима предварительная обработка данных. Примеры такой обработки включают нормализацию [25], фильтрацию [26], понижение размерности [27], сэмплирование и удаление аномалий [28].

2.3. Вывод из главы

Задача рекомендации каналов с использованием данных разных типов, насколько известно автору, никогда не решалась ранее. Применение глубоких нейронных сетей к рекомендациям является трудоёмким и далеко не всегда выгодным способов решения проблемы. Поэтому эта работа фокусируется лишь на более простых в реализации (и дальнейшего развёртвывания) подходах, таких как Slack-алгоритм, основанная на пользователях коллаборативная фильтрация и матричная факторизация.

3. Сбор данных

На данный момент, большинство платформ, которые используются программистами на ежедневной основе, сконцентрированы на одном виде взаимодействия пользователей. Например, Slack [4] дает возможности социального общения посредством текстовых сообщений и звонков, а GitHub [6] хранит исходный код проектов пользователей и помогает их профессиональной коллаборации, предоставляя функциональности для совместной работы над общим проектам, а также рецензирования кода. Получается, что интересные для данной работы социо-технические данные обычно распределены по нескольким платформам. Это сильно затрудняет процесс их сборки, ведь с ростом числа систем число действий, необходимых для их взаимных интеграций, растёт квадратично.

По этой причине, алгоритмы социо-технической поддержки используют только тот тип данных, что напрямую связан с их областью работы (информация из гит репозиториев для подбора рецензентов кода [8]). Более того, такие алгоритмы имеют высокую коммерческую ценность для платформ, следовательно, многие исследования в этой области не являются публичными. Например, Slack имеет свою систему рекомендации каналов, однако, единственная их публикация об алгоритме, лежащем в её основе, датирована 2016 годом [9].

Для данной работы описанная ситуация означала, что не существует в открытом доступе датасетов, которые объединяли бы в себе данные нескольких типов. Более того, данные о каналах (переписки пользователей) являются высоко-чувствительными и не выкладываются для публичного использования. Это значит, что датасет для тренировки рекомендательной системы нужно было собирать самостоятельно. В итоге, датасет был собран с платформ Space [10], Github [6] и Slack [4], причем в датасет были включены только публичные данные (то есть приватные каналы и репозитории пользователей отсутствовали), которые

можно разделить на три модальности: каналы, технические репозитории и структура организации.

Как упоминалось ранее, данные о социальном взаимодействии пользователей и исходный код коммерческих продуктов являются чувствительными, что влечёт за собой некоторое количество проблем и требований, относящихся к безопасности и защите персональных данных. Во-первых, все данные, речь о которых идёт далее в главе, являются публичными внутри организации (то есть датасет не включал в себя приватные каналы и репозитории). Во-вторых, в процессе сбора и обработки данных был принят ряд мер по минимизации собираемых данных и их безопасному хранению. Например, для защиты от потенциального взлома, данные на промежуточных стадиях работы были анонимизированы.

Дальнейшая структура данной главы следующая: в разделе 3.1 подробно описаны данные, полученные с платформы Space, и их потенциальная польза для систем рекомендации каналов. В конце раздела приведено объяснение, почему появилась необходимость расширить датасет с помощью GitHub и Slack. В разделах 3.2 и 3.3 описываются данные, полученные с GitHub и Slack соответственно, а также действия для их интеграции со Space и возникшие в процессе трудности.

3.1. Данные Space

Space [10] — это корпоративная платформа для командной работы, разработанная компанией JetBrains. Она даёт организациям возможность предоставления их сотрудникам общего рабочего цифрового пространства, изолированного на уровне всей компании. В этом пространстве пользователи могут воспользоваться различными функциональностями. Для программной разработки предоставляется гит-хостинг и возможность рецензирования кода. Менеджмент на уровне команд и пользователей возможен с помощью командных

и личных профилей и календарей с расписанием и встречами сотрудников. Это позволяет пользователям, например, планировать встречи и подавать информацию о предстоящих отпусках, отсутствии по болезни и работе из дома. Для социального взаимодействия пользователи могут воспользоваться каналами — групповыми чатами, в которых сообщения сгруппированы по веткам (тредам) для удобства поиска нужной беседы, а сам чат сконцентрирован на конкретном проекте, теме или команде. Также Space предлагает пользователям распространять информацию внутри своих команд и всей компании посредством текстовых документов, планировать свои проекты списками задач, следить за работой над исправлением ошибок в продуктах и многое другое.

В Space есть открытый программный интерфейс приложения (application programming interface / API), позволивший мне самостоятельно скачать все доступные публичные данные о проектах и структуре организации из рабочего пространства, в котором я состою (пространство организации JetBrains). Однако, так как платформа новая, то в момент сбора данных не было полноценной документации для работы с API, что усложнило процесс работы. Более того, при помощи открытого API нельзя получить данные о каналах. Таким образом, для скачивания разных частей датасета было построено два пайплайна, с открытым и закрытым API, для чего потребовалось сотрудничать с командой Space.

Далее приведено описание данных, которые были объединены в итоговом датасете. Первая модальность — каналы. Каждый канал является набором текстовых сообщений, сгруппированных по тредам, в которых известен текст, автор и время публикации. Дополнительно, доступен список подписчиков канала, его название и описание (последнее не гарантировано). Эта информация является основной для алгоритмов рекомендации каналов, что позволило построить несколько унимодальных систем, лежащих в основе общей мультимодальной системы и позже использованных для её валидации. Более подробно эти унимодальные системы разобраны в следующей главе 4 (Система рекомендации каналов), например, алгоритм, предлагающий каналы, в которых есть много

подписчиков, знакомых пользователю по другим каналам. Следующей полученной модальностью данных является структура организации. Она представлена позицией каждого сотрудника в компании (старший исследователь), идентификатором их менеджеров и команд, а также информацией о том, является ли сотрудник главой команды. Стоит обратить внимание, что сотрудники могут принадлежать более, чем одной команде, иметь несколько ролей и менеджеров. Информация о позиции пользователя в компании может форсировать рекомендации, более сфокусированные на схожести места и роли в организации пользователя и подписчиков каналов (раздел 4.4, Эмбеддинг Node2Vec). И последняя модальность — технические репозитории. Из Space я получила репозитории со списком их авторов, ссылкой, по которой можно скачать исходный код, список меток (темы проекта), названием и описанием репозитория (последнее не всегда предоставлено). Данная информация даёт понимание о профессиональных интересах и экспертизах пользователей и может инициировать более разнообразные рекомендации.

После того как все необходимые данные были скачаны, стала очевидна проблема с их количеством. Space — новая платформа, которая была запущена в ограниченный публичный доступ в конце 2019 года, то есть число её пользователей ещё только растет. Для данной работы это значит, что хоть у меня на руках и оказалась полная структура организации, но количество данных по использованию каналов и репозиториев было сравнительно небольшим. Как уже было упомянуто, эти данные являются чувствительными, поэтому я не могу привести статистику с точными числами, но порядок объёма данных был следующим. Всего, на момент написания диплома, с помощью Space удалось получить информацию о порядка двух тысяч пользователей со списком их принадлежности к командам, ролью и менеджером. Меньше трети сотрудников было подписано хотя бы на один канал, а хотя бы три подписки имело менее десяти процентов от общего числа. В среднем, на каждого пользователя приходилось по две подписки. Что касается каналов, то их оказалось порядка

двухсот штук, не пустыми и активными за последние три месяца при этом являлось около половины. В среднем, на каждый канал приходилось по тридцать подписчиков. Напоследок, из Space было получена информация о примерно пятистах проектах, содержащих порядка двух тысяч репозиториев. В среднем, каждый пользователь работал с четырьмя репозиториями.

Space даёт доступ к большому числу разнообразной информации о социально-технических взаимодействиях пользователей, именно поэтому изначально исследование было сфокусировано на ней. Однако, это новая платформа, и хотя у неё есть большой потенциал, на момент данной работы полученных с её помощью данных оказалось недостаточно. По этой причине, было принято решение дополнить собранный датасет данными о репозиториях сотрудников JetBrains из Github и публичных каналах из Slack.

3.2. Данные GitHub

GitHub [6] — это крупнейший веб-сервис для хостинга IT проектов и их совместной разработки. С его помощью я дополнила данные о репозиториях пользователей Space организации JetBrains. У GitHub, как и у Space, существует открытый API (однако, с подробной документацией), что позволило самостоятельно с минимальными трудностями скачать интересующие данные. По итогу, через GitHub я получила всю ту же самую информацию о репозиториях, что и от Space, кроме привязанных к их проектам списков меток.

Сопоставление со Space датасетом проводилось через сопоставление адресов электронной почты авторов GitHub со списком адресов, существующим у каждого пользователя Space. К сожалению, некоторые сотрудники JetBrains оказались ещё не зарегистрированы в Space, а некоторые используют для работы с GitHub email-адреса, не указанные в Space. Поэтому часть авторов некоторых репозиториев осталась не определенной.

Данные с GitHub позволили дополнить данные, полученные со Space, порядком трёх тысяч репозиториев, что увеличило датасет репозиториев в два раза.

3.3. Данные Slack

Slack [4] — самый популярный на данный момент корпоративный мессенджер, основанный на системе каналов. С его помощью я планировала дополнить данные о каналах Space. В то время как из Space удалось получить около двухсот публичных активных каналов, в Slack пространстве организации JetBrains существует порядка тысячи публичных активных каналов. Это связано с тем, что Space является новым продуктом, только начинающим входить в ежедневный процесс взаимодействия сотрудников, в то время как Slack используется организацией уже несколько лет.

Однако, на этом этапе я столкнулось с проблемой отсутствия у Slack открытого API. Это значит, что я не могла собственными усилиями получить данные о каналах, нужно было подавать запрос на доступ к ним. Как уже говорилось, данные о взаимодействии сотрудников в организации крайне чувствительны, в связи с чем при работе с ними требуется принятие мер по исключению возможности несанкционированного доступа, минимизации набора собираемых и хранимых данных до минимально необходимых, а также системная оценка рисков, связанных с безопасностью и защитой персональных данных.

Для удовлетворения этим требованиям, перед получением данных были проведены процедуры по обеспечению их безопасного хранения и обработки. Среди этих процедур были консультации с коллегами, ответственными за безопасность информации и защиту данных, по вопросам безопасного сбора, обработки и хранения данных, организация безопасного окружения для работы с данными, а также процедура оценки рисков приватности (Privacy Impact Assessment, PIA) по стандарту ISO 27005 [29].

3.4. Выводы по главе

В результате данного этапа работы был собран датасет публично доступных взаимодействий порядка двух тысяч сотрудников компании JetBrains с их командами, позициями и менеджерами. Он состоит из информации о примерно пяти тысячах репозиториев и двухстах каналах. В данный момент всё ещё идёт процесс получения данных Slack о публичных каналах пространства JetBrains. Эти данные позволят расширить модальность каналов в датасете и, возможно, лучше обучить систему рекомендации каналов.

Как было отмечено ранее в этой главе, большинство данных, которые доступны в Space и Slack, являются чувствительной информацией из-за рисков, связанных с приватностью и коммерческой тайной. В связи с этим было проведено несколько процедур по обеспечению безопасности системы их хранения и обработки. Стоит отметить, что приведение процедуры сбора и хранения данных в соответствие с внутренними стандартами и регуляторными документами составило существенную часть работы.

4. Система рекомендации каналов

Система рекомендации каналов — алгоритм социо-технической поддержки, позволяющий расширить взаимодействия сотрудников друг с другом, что ведёт к появлению новых связей внутри организации. Как следствие, может возрасти число меж-командных проектов и увеличиться автобусный фактор. Автобусный фактор — мера, показывающая сколько человек, которое должно по какой-либо причине “исчезнуть”, чтобы работа над некоторым проектом остановилась из-за нехватки компетентных информированных сотрудников.

Существует несколько основных подходов к построению рекомендательных систем, два из которых были реализованы в данной работе: основанная на пользователях коллаборативная фильтрация [11] (раздел 4.2) и матричная факторизация матрицы близости сущностей (каналов) и пользователей [30, 31, 32] (раздел 4.3). Матричная факторизация даёт представления пользователей и сущностей, и их потенциальное взаимодействие (подписка) оценивается какой-то функцией схожести, например, косинусным расстоянием. По этому принципу можно построить рекомендации, основанные на представлениях пользователей и каналов, полученных любым другим методом. Поэтому ещё одной реализованной в моей работе унимодальной системой была модель с применением Node2Vec эмбеддингов [33] для графов взаимодействия пользователей (раздел 4.4). И, наконец, я также реализовала подход к рекомендации каналов, описанный Slack [9].

Всю собранную в процессе статистику я использовала для построения решающих деревьев — итоговой системы рекомендации каналов с применением информации о разных типах данных. Итоговый полный список факторов представлен в Таблице 4.1 ниже. Стоит обратить внимание, что полученная статистика является довольно общей. То есть её использование не ограничено системами рекомендации каналов, её (дополненную с фокусом на новую задачу)

можно будет применить для построения других алгоритмов социо-технической поддержки, например, системы рекомендации рецензентов кода.

Таблица 4.1. Полный список факторов, собранных в процессе обработки данных.

	Channel	Structure	Repositories
	#subscribers, #subscriptions, activity-[thread|message]-2021-05-01, activity-[thread|message]-2021-02-01, activity-[thread|message]-2020-11-01, activity-[thread|message]-2020-05-01		bm25-repository2channel, bm25-channel2repository
User Based Collaborative Filtering	cf_channel, cf_[thread|message|mention]_2020-05-01, cf_[thread|message|mention]_2020-11-01, cf_[thread|message|mention]_2021-02-01, cf_[thread|message|mention]_2021-05-01	cf_team, cf_role, cf_team_role	cf_repository
Matrix Factorization	[bpr|als|lmf]_channel, [bpr|als|lmf]_[thread|message]_2020-05-01, [bpr|als|lmf]_[thread|message]_2020-11-01, [bpr|als|lmf]_[thread|message]_2021-02-01, [bpr|als|lmf]_[thread|message]_2021-05-01		[bpr|als|lmf]_repository
Node2Vec	n2v_channel, n2v_[thread|message]_2020-11-01	n2v_struct	n2v_repository
Slack	slack_[thread|message]_2020-05-01, slack_[thread|message]_2020-11-01, slack_[thread|message]_2021-02-01, slack_[thread|message]_2021-05-01		

4.1. Статистика

а) Взаимодействие пользователей с сущностями.

В данном типе статистики строились матрицы, показывающие степень взаимодействия каждого пользователя с каждой сущностью (каналы и репозитории).

Для каналов были построены три типа матриц, различающихся уровнями взаимодействия. Во-первых, уровень канала (далее эта матрица обозначена как *user2channel_channel*) показывает, подписан ли рассматриваемый пользователь на рассматриваемый канал. Во-вторых, уровень тредов (далее эта матрица обозначена как *user2channel_thread*) показывает, в каком проценте тредов участвовал подписанный на канал пользователь. Считается, что пользователи участвовал в треде, если они написали в него хотя бы одно сообщение. Наконец, уровень сообщений (далее эта матрица обозначена как *user2channel_message*), где каждый тред с предыдущего уровня взаимодействия взвешен по количеству

сообщений, отправленных пользователем. Все формулы подсчета весов матриц представлены ниже:

$$user2channel\_channel[us][ch] \;=\; us \in S(ch),$$

$$user2channel\_threadl[us][ch] = \frac{\sum_{thr \in Thr(ch)} us \in S(thr)}{|Thr(ch)|} \cdot \; (us \in S(ch)),$$

$$user2channel\_message[us][ch] = \frac{\sum_{thr \in Thr(ch)} \frac{|Mes(thr, us)|}{|thr|}}{|Thr(ch)|} \cdot \; (us \in S(ch)),$$

где *S(ch)* — список подписчиков канала *ch, Thr(ch)* — список тредов канала *ch* и *S(thr)* — список участников треда *thr*, *Mes(thr, us)* — список сообщений пользователя *us* в тред *thr*.

Аналогичные матрицы планировалось построить для репозиториев, где уровни взаимодействий были бы следующими: само наличие пользователей среди авторов репозитория, далее взвешенное последовательно на процент веток, коммитов в ветках, файлов в коммитах и строк в файлах, в которых участвовали пользователи. Однако, в рамках данной работы я не обрабатывала коммиты, сделанные в репозитории, поэтому для этой модальности получился лишь один уровень взаимодействия:

$$user2rep\_repository[us][repo] \;=\; us \in S(repo),$$

где *S(repo)* — список авторов репозитория *repo*.

б) Взаимодействие пользователей друг с другом.

В данном типе статистики строили матрицы, показывающие степень взаимодействия каждого пользователя с коллегами. Это взаимодействие рассматривалось на всех трех модальностях: каналов, структуры организации и репозиториев.

Как и при взаимодействии пользователей с каналами, для оценивания уровня связи двух пользователей на модальности каналов были построены несколько типов матриц, различающихся уровнями взаимодействия. Во-первых, уровень канала (далее эта матрица обозначена как *user2user_channel*) подсчитывает количество каналов, на которые подписано сразу оба пользователя, по отношению к количеству каналов целевого пользователя. Во-вторых, уровень тредов (далее эта матрица обозначена как *user2user_thread*) показывает, в каком проценте тредов участвовали оба пользователя, по отношению к количеству тредов в канале. Считается, что пользователи участвовали в треде, если они оба написали в него хотя бы по одному сообщению. В-третьих, уровень сообщений (далее эта матрица обозначена как *user2user_message*), где каждый тред с предыдущего уровня взаимодействия взвешен по суммарному количеству сообщений в треде от двух пользователей по отношению к размеру всего треда. И, наконец, уровень упоминаний, где считается число раз целевой пользователь упомянул своего коллегу в сообщениях треда. Все формулы подсчета весов матриц представлены ниже:

$$user2user\_*\ [us_1][us_2] = \frac{\sum_{ch \in C(us_1)} w\_*(us_1, us_2, ch)}{|C(us_1)|},$$

$$w\_channel(us_1,\ us_2,\ ch) = \{us_1, us_2\} \in S(ch),$$

$$w\_thread(us_1,\ us_2,\ ch) = \frac{\sum_{thr \in Thr(ch)} \{us_1, us_2\} \in S(thr)}{|Thr(ch)|} \cdot\ (\{us_1, us_2\} \in S(ch)),$$

$$w\_message(us_1,\ us_2,\ ch) = \frac{\sum_{thr \in Thr(ch)} \frac{|Mes(thr, us_1)| + |Mes(thr, us_2)|}{|thr|}}{|Thr(ch)|} \cdot\ (\{us_1, us_2\} \in S(ch)),$$

$$w\_mention(us_1,\ us_2,\ ch) = \frac{\sum_{thr \in Thr(ch)} \frac{|Ment(thr, us_1, us_2)|}{|thr|}}{|Thr(ch)|} \cdot (\{us_1, us_2\} \in S(ch)),$$

где *C(us)* — список каналов, на которые подписан пользователь *us*, *Ment(thr, us_1, us_2)* — список упоминаний пользователя us_2 пользователем us_1 в треде *thr*.

В дополнение к этому, для всех уровней, кроме каналов, было сделано дополнительно по матрице, где веса считались по тому принципу, однако в учёт шли только те треды, которые инициировал целевой пользователь, то есть написал первое сообщение из группы. Однако, для процент пар пользователей с ненулевым значением этой метрики оказался очень мал, поэтому от использования таких статистик пришлось пока что отказаться. Позднее, когда данных агрегированных в Space станет больше, возможно, имеет смысл вернуться к такой метрике.

Аналогичные матрицы планировалось построить для модальности репозиториев с уровнями, аналогичными взаимодействию пользователей с репозиториями. Однако, все эти уровни, кроме самого верхнего, тоже будут подсчитаны в будущей работе (раздел 6.2).

$$user2user\_repository[us_1][us_2] = \{us_1,\ us_2\} \in S(repo)$$

Наконец, для модальности структуры были построены матрицы, показывающие схожесть двух сотрудников по их позиции в компании в смысле менеджера, роли, команды и статуса лидерства. Число в матрице (точнее его битовая запись) показывает, какие из перечисленных характеристик у пользователей совпадают. Например, $user2user\_structure[us_1][us_2] == 13$ (1101) говорит, что пользователи оба являются лидерами своих команд, имеют одинаковую роль (например, старший разработчик), одного менеджера, но состоят в разных командах.

$$user2user\_structure[us_1][us_2] = \{bit\_manager,\ bit\_role,\ bit\_team,\ bit\_lead\}$$

в) Прочие характеристики сущностей.

Перечисленные выше матрицы строились для того, чтобы облегчить процесс построения систем рекомендации каналов из следующих разделов. Помимо этого, был собран набор факторов, характеризующий сущности и их взаимодействие.

Характеристика каналов**:**

- *#subscribers* — количество подписчиков в канале
- *#subscriptions* — число подписок у пользователя
- *channel_activity_threads_since_{date}/channel_activity_messages_since_{date}* — сколько сообщений/тредов было написано в канал, начиная с какой-то даты. В качестве даты использовалось время за год, полгода, 3 месяца и месяц до скачивания данных. Эта характеристика нужна, чтобы не рекомендовать пользователям “мёртвые” каналы, которые могли быть популярны ранее, но на момент построения рекомендаций являются заброшенными.

Характеристика схожести каналов и репозиториев:

- *tags_intersection* — количество пересечений между пятью самыми встречающимися метками на репозиториях пользователя с набором, построенным из трех самых встречающихся меток каждого подписчика в канале. Этот фактор планировался для сопоставления тематик репозиториев и каналов. Однако, в данных, полученных с GitHub, в отличие от Space, меток у репозиториев нет. А так как больше половины репозиториев получены именно с GitHub, то данный фактор пришлось переосмыслить в следующие два фактора.
- *BM25_repository2channel* — Okapi BM25 метрика, у которой в качестве запроса используется название репозитория, а в качестве корпуса документов для поиска — набор названий каналов, соединенных с их описанием.

- *BM25_channel2repository* — Okapi BM25 метрика, у которой в качестве запроса используется название канала, а в качестве корпуса документов для поиска — набор названий репозиториев, соединенных с их описанием.

Okapi BM25 — это функция ранжирования, используемая в поисковых системах для оценки релевантности документов данному поисковому запросу [32]. Она рассчитывается из частоты встречаемости каждого слова в запросе в рассматриваемых документах по следующей формуле:

$$bm25(D,Q) = \sum_{q \in Q} IDF(q) \cdot \frac{f(q,D)\cdot(k+1)}{f(q,D) + k\cdot(1-b+b\cdot|D| \,/\, avgdl}\,,$$

$$IDF(q) = log[\,(N - n(q) + 0.5) \,/\, (n(q) + 0.5)\,],$$

где *D* и *Q* — документ и запрос соответственно, *f(q, D)* — частота слова *q* в документе *D*, *avgdl* — средняя длина документов в корпусе, *k* и *b* — свободные коэффициенты, *N* — общее число документов в корпусе, а *n(q)* — число документов, содержащих слово *q*. Используя метрику Okapi BM25, я оцениваю связь репозиториев и каналов, согласно их описаниям.

Итого, в результате данного этапа работы был получен набор факторов, характеризующих сущности и их взаимосвязи. Эти факторы не специфичны для задачи рекомендации каналов, они могут быть переиспользованы при построении других алгоритмов социо-технической поддержки (раздел 6.2).

4.2. Основанная на пользователях коллаборативная фильтрация

Логичным подходом оценивания релевантности канала для пользователя является приближение их “схожести” через подписчиков данного канала. Чем больше пользователь похож на других подписчиков, тем больше вероятность, что канал его заинтересует. Похожесть пользователей уже была посчитана ранее матрицами виде *user2user_**. Итоговая релевантность канала для пользователя рассчитывается по следующей формуле:

$$CF\_user2channel\_ * [u_1][ch] = \frac{\sum_{u_2 \in S(ch)} user2user\_*[u_1][u_2]}{|S(ch)|},$$

где * соответствует уровню в одной из трёх модальностей (четыре уровня в каналах, один в репозиториях и три в структуре, описанные далее в разеделе).

Заметим, что для модальностей каналов статистики уже были усреднены по числу подписок у целевого пользователя. Это важно, так как пользователи по-разному пользуются каналами. Кто-то подписывается на много каналов, не сильно отслеживая то, как часто они своими подписками пользуются, а кто-то подписывается лишь на маленькое число самых важных каналов, периодически посещая другие, но оставаясь “внешними” читателями. Более того, как уже было сказано, данных о подписках в собранном датасете очень мало, большинство пользователей ещё не начало пользоваться платформой для общения. По той же причине для модальности репозиториев оценки были взвешены на общее число репозиториев, с которыми успели поработать пользователи.

Формула выше очевидным способов применяется для модальностей каналов и репозиториев. Структура же организации (*user2user_structure*) в себе хранила не вес, а маску схожести пользователей — {bit_manager, bit_role, bit_team, bit_lead}. Для применения в коллаборативной фильтрации, модальность структуры далее разбивается на несколько уровней совпадения позиции пользователей: одинаковая роль (*1**), одинаковая команда (**1*), одинаковая роль и команда (*11*). Для каждого уровня, совпадение маски уровня со схожестью пользователей давало единичный вес. В итоге коллаборативная фильтрация показывала, какой процент подписчиков в канале имеют позицию, совпадающую с позицией целевого пользователя в точности до уровня. Обозначим матрицы с результатом работы основанной на пользователях коллаборативной фильтрации на уровне структуры как *user2user_team, user2user_role, user2user_team-role*. Результаты работы данных моделей представлены в Таблице 4.2. Процесс тестирования моделей описан в разделе 5.1 (Описание экспериментов).

Таблица 4.2. Результаты предсказания каналов с основанной на пользователях коллаборативной фильтрации.

	Precision@3	MAP	ROC AUC
cf_channel	17	18	43
cf_thread_2020-05-01	18	22	44
cf_thread_2020-11-01	18	21	44
cf_thread_2021-02-01	17	21	43
cf_thread_2021-05-01	14	18	44
cf_message_2020-05-01	17	21	44
cf_message_2020-11-01	16	21	44
cf_message_2021-02-01	16	20	43
cf_message_2021-05-01	13	18	43
cf_mention_2020-05-01	15	17	46
cf_mention_2020-11-01	15	17	45
cf_mention_2021-02-01	15	16	45
cf_mention_2021-05-01	11	14	45
cf_repository	15	18	50
cf_team	19	20	53
cf_role	12	15	46
cf_role_team	22	22	56

Как можно заметить, все модели, построенные на каналах, имеют ROC AUC меньше 50%. Это происходит потому, что полученный для тренировки датасет очень мал, в нем менее двух тысяч подписок, и у половины пользователей остаётся не более одной подписки. Причём многие пользователи с маленьким числом подписок следят за каналами с большой аудиторией. Таким образом, при усреднении большая часть релевантностей становится сильно близкой к нулю, что даёт плохие результаты. Поэтому информации о социальном взаимодействии пользователей из построенного датасета недостаточно для построения хорошей рекомендательной системы данным методом. Можно заметить, что использование информации о структуре организации заметно улучшает результаты. Из этого можно сделать вывод, что использование дополнительных модальностей помогает бороться с проблемой холодного старта для пользователей.

4.3. Матричная факторизация

Матричная факторизация используется для разложение матрицы схожести пользователей с сущностями (каналы и проекты) на их скрытые представления, которые при перемножении будут приближать истинный вес в изначальной матрице. Матрицы схожести пользователей с каналами были построены ранее — *user2channel_**, где * показывает уровень рассматриваемой схожести (подписка на канал, участие в канале на уровне тредов и сообщений). При применении к ним матричной факторизации, мы сразу получаем оценку релевантности канала пользователю.

$$MF\_ * [u][ch] = cos(\, emb(u\ ),\ emb(ch)\ ),$$

где emb — функция перевода пользователя или канала в их латентное векторное представление, а cos — косинусная близость двух векторов, являющаяся стандартным способов сравнения векторных представлений.

Однако, для модальности репозиториев (*user2repository_**) результатом работы алгоритма станет оценка релевантности репозиториев для пользователей. Для приведения её к оценке для каналов, используется среднее значение максимальных схожестей всех подписчиков каналов ко всем репозиториям целевого пользователя:

$$MF\_repositttory[us][ch] = \frac{\sum_{u_2 \in S(ch)} max\,[\sum_{r \in R(u_1)} cos(\, emb(u_2), emb(r)\ )\ ]}{|\, S(ch)\ |},$$

где *R(u)* — список репозиториев, в работе над которыми поучаствовал пользователь *u*.

Для получения векторных представлений пользователей и сущностей использовались три алгоритма матричных факторизаций: Alternating Least Squares (ALS) [30], Bayesian Personalized Ranking (BPR) [31] и Logistic Matrix Factorization (LMF) [32]. Лучшие результаты были получены с помощью Bayesian Personalized Ranking, поэтому для построения итоговой модели и дальнейшего сравнения с ней использовались оценки релевантности, полученные именно этим способов. В

Таблице 4.3 представлены все результаты, полученные моделями. Процесс тестирования моделей описан в разделе 5.1 (Описание экспериментов).

Таблица 4.3. Результаты предсказания каналов с помощью матричной факторизации.

	Precision@3	MAP	ROC AUC
bpr_channel	37	27	77
bpr_thread_2020-05-01	30	23	76
bpr_message_2020-05-01	29	23	74
bpr_repository	12	16	40
als_channel	27	21	61
als_thread_2020-05-01	24	20	57
als_message_2020-05-01	22	18	57
als_repository	13	16	45
lmf_channel	29	22	64
lmf_thread_2020-05-01	25	21	61
lmf_message_2020-05-01	24	17	59
lmf_repository	10	16	43

Матричная факторизация требует меньше данных для обучения (для небольшого числа пользователей и каналов, имеющихся у нас, хватает довольно маленькой размерности обучаемых латентых представлений), поэтому результаты её работы значительно лучше, чем у всех других моделей. Bayesian Personalized Ranking показывает результаты лучшие, по сравнению с Alternating Least Squares и Logistic Matrix Factorization моделями, поэтому именно этот алгоритм используется далее для сравнения.

4.4. Векторизация с помощью Node2Vec

Node2Vec — алгоритм, позволяющий извлекать векторные представления вершин во взвешенном связном графе [33]. Это делается по принципу Word2Vec [34], который с помощью softmax функции подбирает представления слов такие, чтобы косинусная смежность двух слов давала результат, близкий к вероятности их совместной встречаемости в тренировочном текстовом корпусе. В случае же

графов последовательность вершин можно сгенерировать с помощью случайных блужданий. Вероятность перехода между вершинами зависит от веса ребра между ними.

Для построения связного графа вершин использовались матрицы схожести пользователей *user2user_**, где * — уровень в одной из модальностей. За оценку релевантности канала для пользователя в результате бралось среднее всех схожестей построенных представлений целевого пользователя и подписчиков рассматриваемого канала.

$$n2v\_*\ [u_1][ch] = \frac{\sum_{u_2 \in S(ch)} cos(emb(u_1), emb(u_2))}{|\,S(ch)\,|}.$$

Таким образом, для модальностей каналов и репозиториев были получены представления на основе социального и технического взаимодействия пользователей. Для модальности структуры строился граф, где вес ребра зависел от связности их позиций в организации. Точнее, веса распределялись по следующим правилам:

- $e(u_1, u_2) = 3$, если у пользователей совпадает команда и роль в команде,
- $e(u_1, u_2) = 2$, если пользователь u_1 является менеджером пользователя u_1,
- $e(u_1, u_2) = 1.5$, если у пользователей совпадает команда,
- $e(u_1, u_2) = 1$, если у пользователей совпадает роль.

4.5. Slack-подобный алгоритм

Алгоритм рекомендации каналов, опубликованный Slack [9] в 2016 году, является вариантом коллаборативной фильтрации, основанной на схожести сущностей. В описанном подходе матрица релевантности каналов пользователям строится на основе времени их активности. Точнее, в качестве оценки используется отношение времени, проведенного за написанием сообщений в канал, ко времени, которое пользователь провёл за чтением канала. Далее полученная матрица используется в качестве представления каналов, с её

помощью строится матрица их схожести (для оценки близости каналов используется косинусная близость их векторов). Каждому каналу сопоставляется фиксированное число каналов (5), самых близких к нему по полученной оценке. Последним этапом, пропуски в изначальной матрице релевантности заполняются как среднее по схожести целевого канала с близкими к нему, взвешенное на релевантность этих каналов для целевого пользователя.

В моей работе были использованы данные с платформы Space, в них не было возможности напрямую узнать статистику по времени активности пользователя в канале. Поэтому, при построении аналогичной модели, я заменила временную активность пользователей на их активность в терминах тредов и сообщений (*user2channel_thread*, *user2channel_message*). Эти активности были подсчитаны за разные промежутки времени (год, полгода, три месяца, месяц). Однако, лучшие результаты показали модели, основанные на активности пользователей за последний год (Таблица 4.5). Именно они использовались для оценки итоговой модели. Процесс тестирования моделей описан в разделе 5.1 (Описание экспериментов). По смыслу алгоритм Slack похож на основанную на пользователях коллаборативную фильтрацию, поэтому неудивительно, что их результаты оказываются схожи.

Таблица 4.5. Результаты предсказания каналов с помощью Slack-подобного алгоритма.

	Precision@3	MAP	ROC AUC
slack_thread_2020-05-01	18	17	49
slack_thread_2020-11-01	17	17	47
slack_thread_2021-02-01	15	16	47
slack_thread_2021-05-01	10	14	43
slack_message_2020-05-01	16	16	46
slack_message_2020-11-01	14	15	46
slack_message_2021-02-01	12	15	44
slack_message_2021-05-01	10	13	42

4.6. Выводы по главе

Итого, в результате было построено четыре вида системы рекомендации каналов, для каждой из которых было обучено несколько моделей на разных модальностях и уровнях взаимодействий пользователей с сущностями. В результате было получено более пятидесяти способов оценки близости пользователей и каналов. Однако, большая часть этих способов давала низкие результаты, поэтому не была использована для итоговых оценок и сравнений.

Основанная на пользователях коллаборативная фильтрация модальности каналов (таблица 4.2) при тестировании на исторических данных продемонстрировала ROC AUC, близкой (или даже ниже) 50%, что значит, что система совершенно не обучена. Это, вероятнее всего происходит потому, что данных о социальном взаимодействии пользователей очень мало (менее двух тысяч подписок), и заметная часть пользователей имеет не более двух подписок, что создает проблему холодного старта. Те же редкие подписки, что имеются у неактивных пользователей, являются крупными каналами вроде общего блога компании (#general). Таким образом, усредненные вероятности схожести пользователей и каналов оказываются близки к нулю. При этом коллаборативная фильтрация на других модальностях даёт несколько лучшие результаты, так как, хоть проблема холодного старта и сильного усреднения остаётся, но данные по этим модальностям гораздо более полные. Итого, для последующего построения системы рекомендации каналов на мультимодальных данных использовались оценки релевантности пользователей и каналов, полученные коллаборативной фильтрацией на уровнях репозиториев и структуры.

Поскольку Slack алгоритм по идее очень близок к реализованной коллаборативной фильтрации, то показываемые им результаты тоже оставляют желать лучшего. Оценки, полученные этими моделями, не были использованы для итоговой системы. В будущем, когда я получу данные о каналах со Slack и объединю их с имеющимся датасетом, то имеет смысл снова попробовать

воспользоваться алгоритмами, от которых в данной работе пришлось отказаться, ведь обученные на большем объёме данных они, вероятнее всего, покажут значительно лучшее качество.

Среди матричных факторизаций лучшие результаты были показаны Bayesian Personalized Ranking алгоритмом, обученным на статистике модальности каналов, собранной за последний год. Это происходит потому, что участвующих в оценке объектов (пользователей и каналов), для которых строятся латентные векторные представления, сравнительно мало, поэтому при построении модели можно использовать вектора низких размерностей, которые не требуют большого числа положительных данных для обучения, в отличие от описанной выше коллаборативной фильтрации. В результате анализа, было принято решение использовать в итоговой системе оценки, полученные Bayesian Personalized Ranking методом матричной факторизации на модальности каналов по статистике активности пользователей за последний год.

Таблица 4.6. Список факторов, используемых при построении мультимодальной системы рекомендации каналов.

	Channel	Structure	Repositories
	#subscribers, #subscriptions, activity-[thread|message]-2021-05-01, activity-[thread|message]-2020-05-01		bm25-repository2channel, bm25-channel2repository
User Based Collaborative Filtering		cf_team, cf_role, cf_team_role	cf_repository
Matrix Factorization	bpr_channel, bpr_[thread|message]_2020-05-01		bpr_repository
Node2Vec		n2v_struct	n2v_repository

Итого, восемнадцать способов оценки использовались в качестве факторов для построения решающих деревьев, представляющих собой итоговую систему рекомендации каналов. Все они перечислены в таблице 4.6. Из них девять получены с использованием лишь данных о каналах (BPR_[channel|thread|message], статистика об активности каналов и пользователей), пять (BM25, CF_repository, BPR_repository, Node2Vec_repository)

основаны на данных, полученных только из модальности технических репозиториев, а четыре — модальности структуры (CF_[team|role|team_role], Node2Vec_structure). На этих факторах было построено четыре вида решающих деревьев, каждый из которых отличался набором модальностей факторов, используемых при построении: каналы, каналы + структура, каналы + репозитории, каналы + структура + репозитории (основная система и результат работы). В следующей главе приведено сравнение результатов работы этих моделей, показанных на тестировании историческими данными, а также их сравнение с лучшими из построенных бейзлайн моделей.

5. Полученные результаты

5.1. Описание экспериментов

Для построения всех систем рекомендаций были использованы только каналы, в которых есть хотя бы 3 подписчика, и которые были активны за последние 3 месяца (к моменту построения систем), то есть в них было опубликовано хотя бы одно сообщение. Это нужно для того, чтобы отфильтровать “мёртвые” каналы, общение в которых сильно ограничено или остановлено, ведь вероятность полезности таких каналов каким-либо пользователям весьма мала. Таким образом, как было упомянуто в разделе 3.1 (Данные Space), в датасете оставалось порядка ста каналов.

Для сравнения построенной системы рекомендации каналов с другими унимодальными моделями (основанная на пользователях коллаборативная фильтрация, матричная факторизация BPR, векторизация Node2Vec, Slack-подобный алгоритм) была проведена оценка на “истории” данных. К сожалению, Space не позволяет узнать точную дату подписки человека на канал, поэтому за время подписки было принято время первого сообщения, написанного пользователем в канал. Каналам же, которые пользователь только читает, были назначены случайные даты подписки.

Целью каждой модели было сделать по три рекомендации каналов для всех пользователей. Поэтому на этапе оценивания использовались только те сотрудники, у которых есть хотя бы три подписки на каналы. Как было упомянуто в разделе 3.1 (Данные Space), после фильтрации в датасете осталось менее двухсот пользователей, которым надо было порекомендовать один из примерно ста каналов. Данные каждого пользователя были разделены на положительные (подписки) и “отрицательные” (остальные каналы). По три штуки самых поздних подписок вместе с тридцатью процентами “отрицательных” данных были использованы в тестовом датасете. Все остальные подписки были объединены с

пятьюдесятью процентами “отрицательных” данных и являлись тренировочными данными для моделей.Стоит отметить, что подписок в полученных для тренировки данных было порядка полутора тысячи штук, что является довольно небольшим числом. А пользователей, у которых меньше двух каналов, оказалось почти половина. Это объясняет, почему алгоритмы коллаборативной фильтрации показали низкие результаты, ведь при усреднении большинство оценок релевантности стали близки к нулю.

В качестве метрик использовалась точность предсказания подписок (из трёх для каждого пользователя), Mean Average Precision и ROC AUC. В следующих разделах показаны результаты проведенных по описанному выше алгоритму сравнений построенной мной общей системы с унимодальными моделями (5.2. Сравнение с бейзлайнами) и сравнение качества итоговой системы в разных вариациях (каналы, каналы + структура, каналы + репозитории, каналы + структура + репозитории) модальностей используемых данных (5.3. Сравнение модальностей). В заключение работы, для уточнения полученных результатов, была проведена оценка рекомендаций пользователями Space, её результаты представлены в разделе 5.4.

5.2. Сравнение с бейзлайнами

Как было упомянуто выше, получившийся для тренировки датасет является довольно небольшим, поэтому следующие результаты, полученные для общей системы на решающих деревьях (‘xgb-all’), не были удивительны (Таблица 5.1). В таблице показаны лучшие результаты, полученные после подбора гиперпараметров. Однако, даже лучшие полученные после подбора гиперпараметров модели (с низким значением ‘sample’ параметров) не смогли избежать переобучения (ROC AUC, близкий к 97% на тренировочных данных при тестировании становится меньше 70%).

Среди всех унимодальных бейзлайнов лучший результат (таблица 5.1) показала матричная факторизация на данных о подписках пользователей на каналы (‘bpr_channel’). Это объясняется тем, что для такого небольшого числа пользователей и каналов, для которых модель обучает векторные представления, их размерность может быть довольно низкой. Следовательно, для обучения матричной факторизации требуется значительно меньше данных, по сравнению с моделями на решающих деревьях.

Также можно заметить, что структурные модели, совершенно не учитывающие активность пользователей в каналах и их тематики, дают сравнительно хороший результат (‘cf_*’). Отсюда можно сделать вывод, что пользователи, занимающие схожие позиции в компании, часто пользуются схожими каналами, а значит структурная модальность должна быть принята во внимание при построении систем рекомендации каналов. Этот вывод подтверждается тем, что у обученных решающих деревьев фактор структурный ‘cf_role_team’ имел значение важности (feature importance), близкое к самому информативному фактору ‘bpr_channel’.

Наконец, модели, построенные на информации о технических репозиториях, тоже дают интересные результаты (‘bm25’). При построении рекомендаций для пользователей (раздел 5.4) я заметила, что решающие деревья на данных о каналах+репозиториях дают более разнообразные рекомендации, чем деревья на данных о каналах+структуре. Это можно объяснить тем, что для большинства сотрудников тематики их репозиториев и проектов довольно разнородны, что при сопоставлении с каналами форсирует выделение таких же разнородных тематик для каналов. Это наводит на заключение, что при построении рекомендательных систем для каналов стоит учитывать данные о технических репозиториях, над которыми работают пользователи, так как они могут помочь с проблемой разнообразия рекомендаций.

Таблица 5.1. Результат сравнения работы итоговой системы на решающих деревьях, обученной на данных всех модальностей, на исторических данных с бейзлайн моделями.

	Precision@3	MAP	ROC AUC
xgb-all	34	25	66
bpr_channel	**37**	**27**	**77**
bpr_thread_2020-05-01	30	23	76
topk_subscribers_size	20	18	44
activity_thread_2020-05-01	11	15	52
cf_role_team	22	22	56
cf_team	19	20	53
cf_role	12	15	46
BM25_repo2channel	21	20	56
cf_repository	15	18	50
bpr_repository	12	16	40

Я предполагаю, что проблема переобучения деревьев и холодного старта у коллаборативной фильтрации решится после дополнения собранного датасета данными о каналах, полученными их Slack. Это косвенно подтверждается тем, что при увеличении размера тренировочных данных (путём отнесения к тестовым данным только двух, вместо трёх, подписок от каждого пользователя, что не только напрямую делает часть положительных данных доступными для обучения, но и позволяет отфильтровывать меньшее число пользователей в начале построения разбиения), тестовый ROC AUC деревьев начинает расти, а тренировочный несколько снижается от своих близких к 100% переобученных значений. К сожалению, на момент написания данного текста, я всё еще ожидаю получения данных со Slack. Как было упомянуто ранее, эти данные являются высоко-чувствительными, поэтому процесс удовлетворения всех требований безопасности оказался довольно долгим. Таким образом, объединение Slack и Space каналов и обучение моделей на расширенных данных стало одной из задач для будущей работы. Однако, чтобы уточнить полученные результаты, было решено провести прямую оценку построенных рекомендательных систем

пользователями Space, результаты которой представлены в разделе 5.4 (Оценка пользователями Space).

5.3. Сравнение модальностей

В таблице 5.2 представлены результаты оценки на исторических данных решающих деревьев, обученных на разных комбинациях модальностей (каналы, каналы + структура, каналы + репозитории, каналы + структура + репозитории). Все эти результаты приведены для лучших моделей, полученных после подбора гиперпараметров. Хоть, как было замечено в предыдущем разделе, все модели и переобучаются, но стоит отметить, что использование всех модальностей помогает бороться с этой проблемой.

Таблица 5.2. Результаты сравнения системы рекомендации на решающих деревьях, обученной на разных комбинациях модальностей, с лучшей бейзлайн моделью (Bayesian Personalized Ranking на подписках).

	Precision@3	MAP	ROC AUC
bpr_channel	37	27	77
xgb-all	34	25	66
xgb-channel+structure	31	23	62
xgb-channel+project	29	21	61
xgb-channel	25	17	57

5.4. Оценка пользователями Space

Я попросила 20 пользователей Space оценить каналы, полученные различными рекомендательными системами: три рекомендации случайных каналов, три рекомендации методом Bayesian Personalized Ranking и три рекомендации, полученных решающими деревьями, обученными на данных всех

модальностей. Итого, каждый участник опроса получил для разметки от 5 до 9 каналов. Разметка производилась следующим образом:

- -1, если канал нерелевантен и неинтересен для пользователя
- 0, если в канале есть релевантная/интересная информация, но подписываться пользователь скорее не стал бы
- 1, если канал релевантен, и пользователь готов на него подписаться.

По результату опроса из шестидесяти случайных рекомендаций 43 были отмечены пользователями как нерелевантные, 13 как интересные, но не стоящие подписки, и 4 канала были отмечены как удачная рекомендация. Это означает, что в среднем участники опроса были довольно критичны и не стали бы подписываться на случайный канал. Полученные результаты представлены на рисунке 5.1.

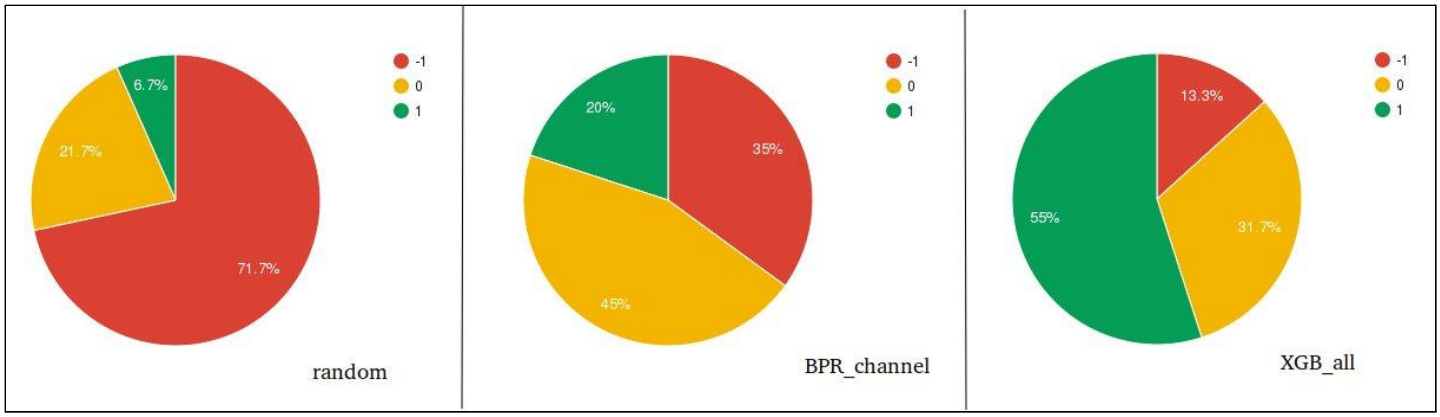

Рисунок 5.1. Результаты оценки рекомендаций моделей пользователями Space.

Из шестидесяти рекомендаций методом Bayesian Personalized Ranking 21 была отмечена как нерелевантная, 27 как релевантная, но недостаточная для подписки, и 12 рекомендаций оказались удачными. Таким образом, метод матричной факторизации на практике показал средние результаты, на большую часть рекомендованных каналов (80%) пользователи отказались подписываться.

И, наконец, из шестидесяти рекомендаций построенной мной системы, работающей на мультимодальных данных, 8 рекомендаций оказались плохими, 19

недостаточно релевантными или интересными, а 33 рекомендации были отмечены пользователями как стоящие подписки. Итого, более половины каналов, рекомендованных пользователям методом, основанным на мультимодальных данных, оказались удачными.

Такое различие результатов тестирования на исторических данных и реальными пользователями можно объяснить тем, что среди участников опроса более половины только начинали пользоваться платформой Space. У них либо совершенно не было подписок на каналы, либо имелось один-два канала, которые пользователь успел изучить. В такой ситуации неудивительно, что матричная факторизация не справилась предоставить хорошие рекомендации. Построенная же мной система воспользовалась информацией о GitHub проектах пользователей и их позиции в компании, и смогла предложить релевантные каналы, справившись с проблемой холодного старта.

6. Заключение

6.1. Результаты работы

В результате проделанной работы был собран датасет, объединяющий в себе три типа данных: каналы, структура организации и технические репозитории. Однако, так как платформа Space, с которой были собраны данные о каналах, является довольно новой, то полученных данных оказалось недостаточно: порядка ста активных каналов и двухсот подписчиков, пользующихся ими. В связи с этим некоторые из реализованных алгоритмов (коллаборативная фильтрация на модальности каналов и Slack-подобная модель) показали низкое качество при тестировании на исторических данных из-за нехватки информации о социальном общении пользователей, то время как система, построенная на решающих деревьев переобучилась на маленьком количестве подписок в тренировочном датасете и дала результаты по точности предсказаний на 3% ниже матричной факторизации на каналах.

Однако, оценка пользователями Space показала, что построенная мной рекомендательная система на данных различных типов даёт предлагает пользователям рекомендации заметно более качественные по сравнению с лучшей из унимодальных моделей — матричной факторизации с помощью Bayesian Personalized Ranking. Это объясняется тем, что многие пользователи, участвовавшие в опросе, не очень активно пользуются Space для социального общения, не имея подписок на каналы, то есть матричная факторизация подверглась проблеме холодного старта пользователей. Откуда можно сделать вывод, что использование разных типов данных при построении рекомендаций помогает решить проблему холодного старта, используя данные о позиции и технических репозиториях пользователей в качестве метрики для сравнения их с коллегами.

6.2. Будущая работа

В данный момент процесс обеспечения безопасности для передачи данных о Slack каналах подходит к концу, и скоро я смогу расширить уже построенный датасет. После чего первым шагом будет обучение всех моделей с нуля. Ожидается, что объёмов социального общения пользователей в новом датасете будет достаточно для более успешного обучения всех моделей.

Ещё одним пунктом работы является более детальный сбор статистики о технических репозиториях пользователей, до уровня активности и взаимодействия пользователей в ветках, коммитах и файлах репозиториев. Такое разбиение на уровни аналогично тому, что уже было проделано для модальности каналов, и оно может дать более чёткое представление о взаимодействии пользователей в процессе разработки проектов и написания кода. Более того, такая статистика будет полезна при построении других алгоритмов социо-технической поддержки, например, рекомендации рецензентов кода.

И, наконец, ещё одной областью для исследования является применение собранной статистики для построения других алгоритмов социо-технической поддержки. Например, рекомендация экспертов. В рабочем пространстве JetBrains (и Slack, и GitHub) существуют каналы, куда пользователи пишут с просьбой посоветовать сотрудника компании, с которым можно было бы проконсультироваться по возникшим у них вопросам и проблемам. Автоматическая система рекомендации таких коллег-экспертов могла бы упростить взаимодействие и работу пользователей Space.

Список литературы

[1] Robillard, M.P., Maalej, W., Walker, R.J., Zimmermann, Th. Recommendation Systems in Software Engineering. 2014, 562 p.

[2] Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Raula Gaikovina K., et.al. Who should review my code? A file location-based code-reviewer recommendation approach for modern code review. // Proceedings of the 22nd International Conference on Software Analysis, Evolution, and Reengineering — SANER ’15 — IEEE Press, 2015.

[3] Valerio Cosentino, Javier Cánovas Izquierdo, Jordi Cabot. Assessing the bus factor of Git repositories. // Proceedings of the 22nd International Conference on Software Analysis, Evolution, and Reengineering — SANER ’15 — IEEE Press, 2015.

[4] Slack [website]. URL: https://slack.com/intl/en-ru/ (date: 20.05.2021)

[5] Telegram Messenger [website]. URL: https://telegram.org/ (date: 20.05.2021)

[6] GitHub [website]. URL: https://github.com/ (date: 20.05.2021)

[7] Google Calendar [website]. URL: https://calendar.google.com/calendar/ (date: 20.05.2021)

[8] H. Alperen Çetin, Emre Doğan, Eray Tüzün. A review of code reviewer recommendation studies: Challenges and future directions. // Science of Computer Programming, Volume 208 — 2021.

[9] Personalized channel recommendations in Slack [website]. URL:

https://slack.engineering/personalized-channel-recommendations-in-slack/
(date: 20.05.2021)

[10] Space [website]. URL: https://www.jetbrains.com/space/ (date: 20.05.2021)

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. // Proceedings of the 10th international conference on World Wide Web — WWW '01 — New York, P. 285–295 — URL: https://doi.org/10.1145/371920.372071

[12] Mining Software Repositories Conference [website]. URL: https://conf.researchr.org/series/msr (date: 20.05.2021)

[13] Automated Software Engineering Conference [website]. URL: https://conf.researchr.org/series/ase (date: 20.05.2021)

[14] Source Code Analysis and Manipulation Working Conference [website]. ULL: http://www.ieee-scam.org/ (date: 20.05.2021)

[15] Vijayarani, S., Ms J. Ilamathi, and Ms Nithya. Preprocessing techniques for text mining - An overview. // International Journal of Computer Science & Communication Networks. — 2015 — Vol. 5, no. 1. — P. 7–16.

[16] Bayardo, Roberto J., and Rakesh Agrawal. Data privacy through optimal k-anonymization. // Proceedings of the 21st International conference on data engineering — ICDE '05 — IEEE, 2005.

[17] Kouters Erik, Vasilescu Bogdan, Serebrenik Alexander, Brand, M.. Who's who in GNOME: using LSA to merge software repository identities. // Proceedings of the 28th

International Conference on Software Maintenance — ICSM '12 — IEEE, 2012.

[18] Goeminne Mathieu, and Tom Mens. A comparison of identity merge algorithms for software repositories. // Science of Computer Programming — 2013 — Vol. 78, no. 8. — P. 971–986.

[19] Siddiqui Tamanna, and Ausaf Ahmad. Data mining tools and techniques for mining software repositories: A systematic review. // Big Data Analytics — 2018 — P. 717-726.

[20] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. A general path-based representation for predicting program properties. // In Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation — PLDI '18 — New York, P. 404–419. URL:https://doi.org/10.1145/3192366.3192412

[21] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. The promises and perils of mining GitHub. // In Proceedings of the 11th Working Conference on Mining Software Repositories — MSR '14 — New York, P. 92–101. URL:https://doi.org/10.1145/2597073.2597074

[22] Thomas Zimmermann, Peter Weisgerber, Stephan Diehl, and Andreas Zeller. Mining Version Histories to Guide Software Changes. // In Proceedings of the 26th International Conference on Software Engineering — ICSE 04 — IEEE 2014, P. 563–572

[23] Vladimir Kovalenko, Fabio Palomba, and Alberto Bacchelli. Mining file histories: should we consider branches? // In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering — ASE '18 — Association for Computing Machinery, 2018, New York, P. 202–213. URL: https://doi.org/10.1145/3238147.3238169

[24] N. Bettenburg, E. Shihab and A. E. Hassan, An empirical study on the risks of using off-the-shelf techniques for processing mailing list data // IEEE International Conference on Software Maintenance, 2009, P.. 539-542,

[25] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. // IEEE Transactions on Nuclear Science, 1997 — Vol. 44, no. 3, P. 1464-1468

[26] Maalej Walid, Thomas Fritz, Romain Robbes. Collecting and Processing Interaction Data for Recommendation Systems. // Recommendation Systems in Software Engineering, 2014, P. 173-197.

[27] Xavier Amatriain, Alejandro Jaimes, Oliver Nuria & Josep Pujol. Data Mining Methods for Recommender Systems. // Recommender systems handbook, 2011, P. 39-71.

[28] Toledo, Raciel Yera, Yailé Caballero Mota, and Milton García-Borroto. A Regularity-Based Preprocessing Method for Collaborative Recommender Systems, 2013

[29] Information technology — Security techniques, ISO 27005 [website]. URL: https://www.iso.org/standard/75281.html (date: 20.05.2021)

[30] Hu Yifan & Yehuda Koren, Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets. Proceedings// International Conference on Data Mining — ICDM., 2008 — IEEE, ‘08, P. 263-272.

[31] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. // In Proceedings of the

Twenty-Fifth Conference on Uncertainty in Artificial Intelligence — UAI '09 — AUAI Press, 2009, Arlington, P., 452–461.

[32] Christopher C. Johnson. Logistic Matrix Factorization for Implicit Feedback Data. // NIPS worksion, 2014

[33] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks, CoRR 2016.

[34] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space // CoRR. — 2013. — Vol. abs/1301.3781. — URL: http://dblp.uni-trier.de/db/journals/corr/corr1301.html# abs-1301-3781

[35] F. Maxwell Harper and Joseph A. Konstan. The MovieLens Datasets: History and Context. // ACM Trans, 2015, 19 pages. DOI:https://doi.org/10.1145/2827872

[36] Benjamin Stark, Constanze Knahl, Mert Aydin, Karim Elish. A Literature Review on Medicine Recommender Systems. // International Journal of Advanced Computer Science and Applications, 2019.

[37] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, et. al. Wide & Deep Learning for Recommender Systems. // In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems — DLRS ‘16 — New York, 2016, P. 7–10. DOI:https://doi.org/10.1145/2988450.2988454

[38] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep Learning Based Recommender System: A Survey and New Perspectives. // ACM Comput, 201. DOI:https://doi.org/10.1145/3285029

[39] Andreas Argyriou, Miguel González-Fierro, and Le Zhang. Microsoft Recommenders: Best Practices for Production-Ready Recommendation Systems. // In Companion Proceedings of the Web Conference 2020 — WWW '20 — New York, 2020, P. 50–51. DOI:https://doi.org/10.1145/3366424.3382692

[40] Schafer J.B., Frankowski D., Herlocker J., Sen S. Collaborative Filtering Recommender Systems.// The Adaptive Web. Lecture Notes in Computer Science, 2007, Vol 4321. URL: https://doi.org/10.1007/978-3-540-72079-9_9