

## **Аннотация**

### **Цели и задачи работы**

- Рассказать об основных задачах в мире графовых нейронных сетей
- Провести теоретический обзор решения задач рекомендательного моделирования с помощью графовых нейросетей
- Провести эксперимент по переобучению выбранной архитектуры на графовых нейронных сетях для задачи рекомендации новостных публикаций пользователю.

### **Полученные результаты**

- Выбранная архитектура действительно смогла переобучиться под другую задачу рекомендации с неплохими результатами по метрикам на тестовой выборке.
- Был проведён дополнительный эксперимент по улучшению качества рекомендательной системы с помощью внедрения предобученных эмбеддингов в архитектуру. Данный эксперимент оказался не таким успешным, тем не менее, небольшое улучшение качества было достигнуто.

**Ключевые слова:** *рекомендательные системы, графовые нейронные сети, рекомендательное моделирование.*

# Оглавление

0.1	Введение . . . . .	4
0.2	Рекомендательное моделирование с помощью графовых нейронных сетей . . . . .	7
0.2.1	Постановка задачи . . . . .	7
0.2.2	Обзор существующих подходов . . . . .	7
0.3	Основная архитектура . . . . .	12
0.3.1	Моделирование пользователей . . . . .	14
0.3.2	Моделирование предметов . . . . .	16
0.3.3	Рейтинговое предсказание . . . . .	16
0.3.4	Обучение модели . . . . .	17
0.4	Вычислительный эксперимент . . . . .	18
0.4.1	Экспериментальные данные . . . . .	18
0.4.2	Структура экспериментальных данных . . . . .	19
0.4.3	Исходные данные . . . . .	22
0.4.4	Предобработка экспериментальных данных . . . . .	24
0.4.5	Тренировка модели . . . . .	26

0.4.6 Тестирование модели . . . . .	29
0.5 Заключение . . . . .	31
0.5.1 Итоги работы . . . . .	31
0.5.2 Дальнейшие исследования . . . . .	32
Список литературы . . . . .	33

# 0.1 Введение

Графовые нейронные сети сегодня решают огромный спектр задач. Прежде всего я хотел бы вкратце рассказать о таких задачах, дабы понимать их важность и актуальность. Затрону следующие:

- Классификация.

Может быть как самой вершины графа, так и графа в целом. Например, имеем на входе граф связанных между собой пользователей (для каждого пользователя храним также дополнительную информацию о его предпочтениях, подписках, интересах; в общем, типичная информация из социальной сети). Задаёмся целью определить, к какой социальной группе относятся данные пользователи: футбольные гики, любители автомобилей, приверженцы здорового образа жизни и так далее. Это и есть задача классификации графа. Если же нам необходимо узнать, к какой группе принадлежит конкретный пользователь на основе принадлежности к той или иной группе его ближайших соседей по графу (либо не только ближайших, но об этом чуть позже), то мы решаем задачу классификации вершины графа.

- Кластеризация.

Как известно, задачи такого типа относятся к обучению без учителя. И они очень актуальны, так как абсолютное большинство любых данных, в том числе в графовом представлении, остаются неразмеченными на сегодня. Кластеризация позволяет структурировать и упорядочить сырье данные, тем самым в разы увеличивая скорость и качество поиска по ним, а также хранить их в удобном виде. Наиболее яркий пример задачи кластеризации в мире графов: обнаруже-

ние объединённых набором общих признаков сообществ, групп пользователей в социальных сетях. Вообще говоря, любые объекты можно кластеризовать: товары из интернет-магазинов, научные статьи и так далее.

- Предсказание связи между объектами.

Такой род задач является наиболее актуальным для рекомендательных систем. К примеру, мы хотим порекомендовать пользователю социальной сети возможного друга, который ещё таковым не является. Либо же рекомендуем пользователю потенциально интересные для него публикации, товары в интернет-магазинах, статьи. Причём важно не только предсказать такую рекомендацию, но и её рейтинг(то есть, насколько мы уверены в актуальности предсказания). На формальном языке графов необходимо ответить на вопрос, можно ли соединить две вершины графа (суть предсказания) и каков будет вес данного ребра (степень уверенности в предсказании).

В качестве темы своей дипломной работы я решил выбрать применение графовых нейросетей для рекомендательных систем. Поэтому, работа посвящена именно задаче предсказания связи между объектами. Я рассмотрел несколько существующих подходов(архитектур) и решил сосредоточиться на одной из них. Научная публикация [1] о данной архитектуре является самой цитируемой на тему "Графовые нейросети для рекомендательных систем". Данный подход не является самым научно-изощрённым, но даёт хороший практический результат и хорошо понятен на идеином уровне. Идея в следующем: предсказать релевантность, актуальность предлагаемого товара пользователю в интернет-магазине. На вход подаётся пара объектов пользователь-товаров, а на выходе - рейтинг важности товара для

пользователя.

С экспериментальной точки зрения, я взял за основу реализацию архитектуры по ссылке [2] - и переобучил её на других данных, о чём можно подробно почитать в основном разделе. Но идеально, я использовал предложенную архитектуру для рекомендации статей пользователю. Теперь уже не товаров, а публикаций на сайте Майкрософт Новости. В силу специфики данных система обучилась предсказывать, понравится читателю статья или нет. То есть многоклассовая классификация (изначально было 8 дискретных значений рейтинга) стала бинарной.

## 0.2 Рекомендательное моделирование с помощью графовых нейронных сетей

### 0.2.1 Постановка задачи

Задача рекомендательного моделирования состоит в том, чтобы спроектировать архитектуру, которая на входе любым из доступных способов обрабатывает информацию об объектах рекомендации; о том, кому рекомендовать; о всевозможных взаимосвязях между объектами. На выходе же подобная архитектура выдаёт либо некую рекомендацию(это может быть любой объект из существующего графа), либо рейтинг совместимости разных объектов (проще говоря, насколько один объект подходит другому в заранее выбранной шкале).

### 0.2.2 Обзор существующих подходов

Как уже было сказано, одной из основных компонент в построении рекомендательных систем является решение задачи link prediction. Далее в этом разделе постараемся описать несколько основных подходов, решающих поставленную задачу.

Одним из наиболее наглядных семейств методов предсказания связи между вершинами графовой структуры является подход, который может быть назван как эвристические методы. Суть таких методов в следующем: будем предполагать, что если для двух рассматриваемых вершин  $x$  и  $y$  существует достаточное количество общих соседей, то с большой вероятностью можно утверждать, что рассматриваемые вершины похожи и что между ними потенциально может быть ребро. А вот дальше будем разли-

чать эвристики по общей формуле определения множества соседей и порядку удаленности соседей от вершин  $x$  и  $y$ .

Как одной из наиболее простых эвристик остановимся на методе common neighbors представлена в статья [3].

$$A(x, y) = |\Gamma(x) \cap \Gamma(y)|,$$

где  $\Gamma(x)$  - множество вершин смежных с  $x$ .

Понятно, что чем больше будет коэффициент  $A(x, y)$ , тем вероятнее увидеть ребро между вершинами. Также видим, что в формуле участвуют только два множества -  $\Gamma(x)$  и  $\Gamma(y)$ , которые являются непосредственными соседями вершин. Поэтому порядок эвристики будет первый.

К методам первого порядка можно отнести также preferential attachment [4]:

$$A(x, y) = |\Gamma(x)| \cdot |\Gamma(y)|,$$

где  $\Gamma(x)$  - множество вершин смежных с  $x$ .

Аналогично, Jaccard коэффициент [5]:

$$A(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|},$$

где  $\Gamma(x)$  - множество вершин смежных с  $x$ .

Существуют методы второго порядка, где рассматриваются множества соседей всех вершин, которые являются соседями для двух начальных, например Adamic-Adar или resource allocation. Формулы для которых можно наблюдать в следующей таблице:

название	формула	порядок
common neighbors	$ \Gamma(x) \cap \Gamma(y) $	first
Jaccard	$\frac{ \Gamma(x) \cap \Gamma(y) }{ \Gamma(x) \cup \Gamma(y) }$	first
preferential attachment	$ \Gamma(x)  \cdot  \Gamma(y) $	first
Adamic-Adar	$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log  \Gamma(z) }$	second
resource allocation	$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{ \Gamma(z) }$	second

Помимо упомянутых выше, существуют эвристики более высокого порядка, но на них останавливаться не будем.

Теперь же хочется перейти к принципиально другому методу решения задачи link prediction, а именно SEAL [6]. В подходе используются графовые нейронные сети, о которых стоит упомянуть отдельно.

В 2016 году Kipf [7] вводит понятие графовой сверточной нейронной сети - GCN. Блок стал одним из самых популярных инструментов в области работы над графовыми структурами. С помощью него можно решать не только задачу link prediction, но и другие, такие как node classification, graph classification. Сама модель принимает на вход матрицу весов узлов рассматриваемого графа и его метрицу смежности. С помощью описанного ниже слоя будут генерироваться представления вершин, при этом после каждого слоя, представления получают дополнительную информацию о соседних вершинах (их представлениях). Формульно, один GCN блок можно описать следующим образом:

$$\mathbf{h}_x^{(l)} = \sigma \left( \mathcal{W}^{(l)} \sum_{y \in \{x\} \cup \mathcal{N}(x)} \tilde{\mathcal{A}}_{x,y} \mathbf{h}_y^{(l-1)} \right), \quad (1)$$

где  $\mathbf{h}_x^{(l)}$  - представление вершины  $x$  после  $l$ -го слоя,  $\mathcal{W}$  - обучаемая матрица

весов,  $\mathcal{A}$  - матрица смежности.

Операция свертки в GCN-сети применяется к каждой вершине по отдельности. Авторы SEAL [6] утверждают, что решение задачи link prediction оптимальнее проводить, когда свертка проводится вокруг каждого ребра. Весь подход SEAL можно разбить на три основные составляющие:

- extracting enclosing subgraph
- node information matrix construction
- convolution for enclosing subgraphs

Остановим на каждом из них по отдельности. Сначала для каждого рассматриваемого ребра нужно выделить небольшую область вершин, которые по предположению смогут посвятить на ответ о наличии ребра. Процесс формирования таких подграфов представлен на рисунке 1.

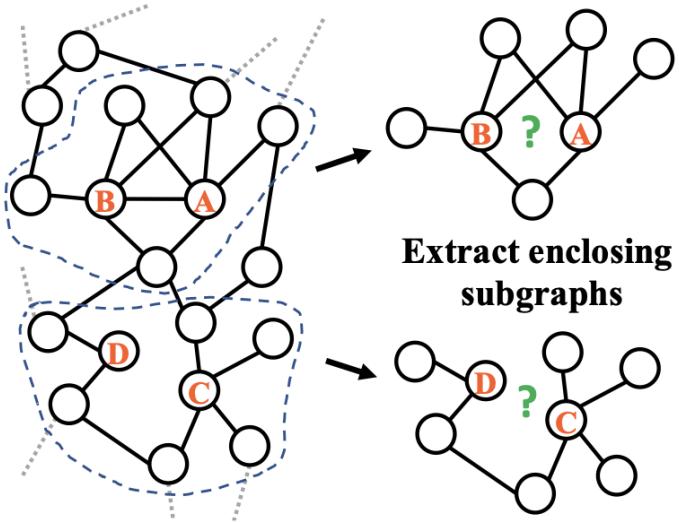


Рис. 1: Выделение подграфа.

Далее для каждого подграфа формируются матрицы признаков вершин. В свою очередь признаки состоят из нескольких компонент: изна-

чальные признаки датасета, обучаемые embeddings и (нововведение статьи) структурные метки, показывающие степень удаленности каждой вершины от ребра рассматриваемого подграфа. Значение метки формирует по следующей незамысловатой схеме - рисунок 2.

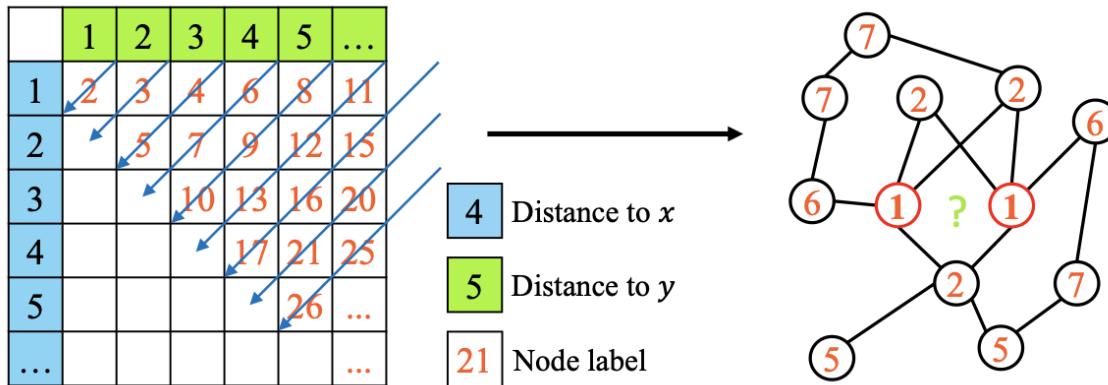


Рис. 2: Формирование структурной метки вершины.

После чего к каждому подграфу может быть применена операция свертки.

Существует множество других подходов в области графовых нейронных сетей, но качественно нам было интересно остановиться именно на SEAL.

### 0.3 Основная архитектура

Как было упомянуто выше, за основу своей экспериментальной части работы я взял архитектуру из статьи [1]. Которую сейчас хотелось бы описать немного подробнее. Насчём с малого: на рисунке ниже представлен простейший граф, отображающий взаимодействие пользователей друг с другом и одного пользователя с релевантными ему предметами. Самый простой способ хранить подобные данные - с помощью словарей в языке Python. Для каждого пользователя (это ключ в словаре) задаётся список из релевантных ему предметов и соответствующих рейтингов(такие списки и есть значения в словаре, получаемые по ключу).

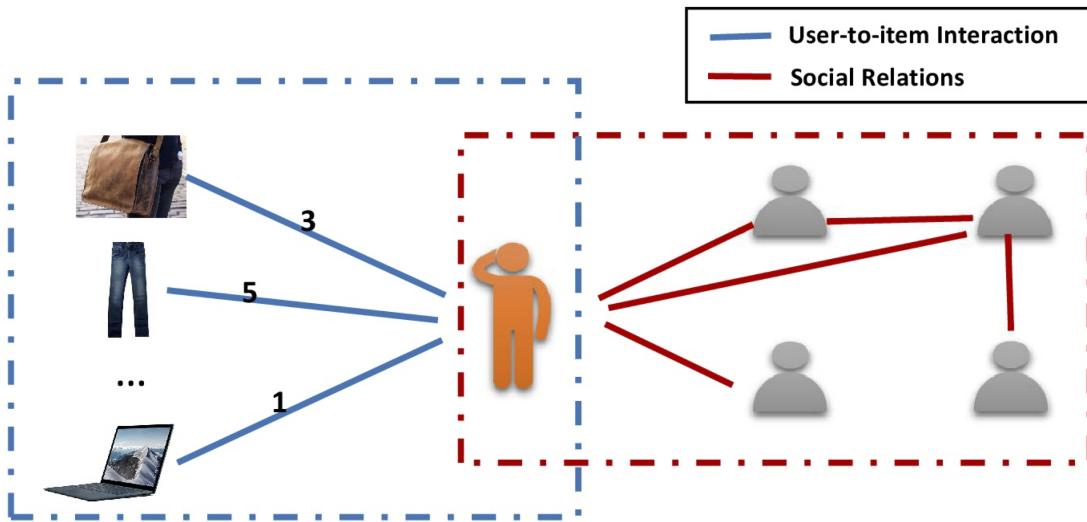


Рис. 3: Графовые данные в социальных рекомендациях. Основа состоит из двух графиков, включая граф пользователь-предмет (левая часть) и социальный граф пользователь-пользователь (правая часть). Обратите внимание, что номер на ребре графа пользователь-предмет обозначает мнение (или рейтинг) предмета по отношению к пользователю.

На рисунке 3 не представлен третий важнейший граф, построенный по аналогичному принципу. В нём для каждого предмета задаётся список

пользователей, которые с данным предметом взаимодействовали, и тот же рейтинг связи предмет-пользователь. Можно увидеть, что в данной архитектуре входные графовые данные по отдельности не являются слишком сложными: графы пользователь-предметы и предмет-пользователи - эвристики 1-го порядка. Это значит, что за один прыжок можно добраться от пользователя к любому предмету и от любого предмета к соответствующему пользователю. Однако в совокупности граф получается очень запутанным, причём в нём наравне рассматриваются пользователи и предметы - объекты одной большой графовой структуры.

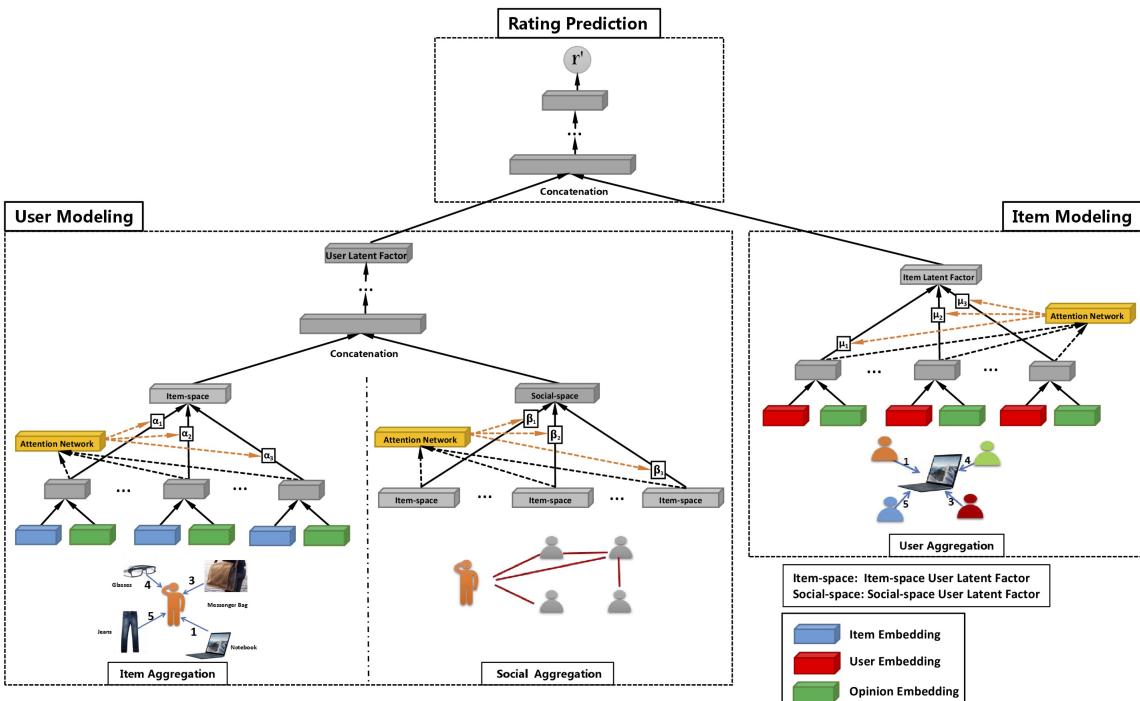


Рис. 4: Общая архитектура предлагаемой модели.

Теперь хотелось бы перейти к самой архитектуре и её подробному описанию. На рисунке 4 можно выделить 3 основных части: моделирование пользователей, моделирование товаров и рейтинговое предсказание. Разберем каждую из трёх частей. Но для начала хотел бы пояснить суть синих,

зеленых и красных параллелепипедов на изображении. Это эмбеддинги - синий соответствует определенному предмету рекомендации, красный - пользователю, зелёный - рейтингу предмета для определённого пользователя. В оригинальной статье данные эмбеддинги генерируются случайным образом и обучаются вместе с другими матрицами весов (это и есть ещё один слой матриц весов размера кол-во пользователей/предметов/рейтингов по итоговой шкале \* размер эмбеддинга). Это значит, что данные эмбеддинги не предобучены и не несут в себе некой реальной информации о взаимном расположении пользователей/предметов в высокоразмерном пространстве(размерность которого равна длине эмбеддинга). В разделе "Эксперименты" я расскажу подробнее о попытке улучшения архитектуры посредством добавления предобученных на реальной информации эмбеддингов предметов.

Однако сейчас не об этом. Суть архитектуры в том, чтобы для заранее заданной пары объектов пользователь-предмет предсказать рейтинг предмета для данного пользователя на основе всей информации о пользователе и предмете, которую мы знаем из графовых структур.

Теперь приступим к рассмотрению трёх основных блоков на рисунке 4.

### 0.3.1 Моделирование пользователей

Самая большая часть архитектуры, отвечающая за генерацию латентных эмбеддингов пользователей. По факту это энкодер, который обучается собрать всю необходимую информацию о пользователе: о его предпочтениях в мире предметов (агрегирование предметов) и о его социальных связях с другими пользователями (социальное агрегирование). Обе этих части также стоит рассмотреть отдельно.

## Агрегирование предметов

В этой секции используется только граф пользователь-предметы. На вход подаётся  $N$  раз дублированный эмбеддинг пользователя (где  $N$  - количество связанных с ним предметов) вместе со всеми эмбеддингами рейтингов, соответствующими своим предметам. Для каждого предмета делается конкатенация эмбеддинга с пользовательским. Далее эмбеддинги проходят через многослойную нейросеть и попадают на алгоритм внимания (Attention). Это ключевая концепция во всей блоках данной архитектуры, поскольку она позволяет учесть вес каждого из предметов для данного пользователя (в соответствии со своими рейтингами, конечно) для того, чтобы наиболее точным образом получить скрытое представление каждого пользователя. Интуитивно это довольно понятно: если мы получим латентное представление пользователя просто путём взятия среднего эмбеддинга по всем предметам, мы никак не учтём рейтинг предмета для пользователя, а именно рейтинг несёт в себе ключевую информацию о предпочтениях. Итого, механизм внимания позволяет обучиться выбирать самую полную информацию о пользователе, что активно используется в следующих блоках.

## Социальное агрегирование

На вход этой части архитектуры подаются те самые латентные эмбеддинги пользователей, выученные в блоке "Агрегирование предметов". Естественно, используются эмбеддинги лишь тех пользователей, кто непосредственно связан с выбранным нами заранее пользователем через социальный граф. В этой части также используется механизм внимания, чтобы получить более точный социальный эмбеддинг пользователя. Опять же, подход

"пользователь - арифметическое среднее связанных с ним пользователей" не учитывает тот факт, что кто-то из пользователей очень близок по предпочтениям к данному и соответственно должен оказывать большее влияние на социальный эмбеддинг данного пользователя.

Итого, два полученных эмбеддинга (латентный эмбеддинг пользователя и его социальный эмбеддинг) конкатенируются, проходят через многослойную нейросеть. В результате получается эмбеддинг(на рисунке 4 он называется User Latent Factor), полностью описывающий пользователя на основе всех связанных с ним графовых данных.

### 0.3.2 Моделирование предметов

В данной части архитектуры происходят аналогичные блоку "Агрегирование предметов" операции, только над определённым предметом (например, целью архитектуры является оценить совместимость заранее заданного пользователя и предмета). Для всей пользователей, связанных с данным предметом, делается конкатенация эмбеддингов пользователя и рейтинга, соответствующего данному предмету. Затем механизм внимания, который позволяет наиболее точно построить вектор описания предмета(Item Latent Factor) в соответствии с тем, как его оценивают разные пользователи.

### 0.3.3 Рейтинговое предсказание

Этот блок архитектуры объединяет вычисленные ранее информативные представления заранее заданных пользователя и предмета, прогоняет через многослойную нейросеть, которая по сути является многоклассовым классификатором и позволяет получить рейтинг совместимости пользова-

теля и предмета. Иными словами, насколько предмет потенциально подойдёт пользователю на основе его предпочтений.

### 0.3.4 Обучение модели

В целом происходит незамысловатым образом. На обучающей выборке число потерь вычисляется методом средней квадратичной ошибки в предсказании рейтинга. На тестовой же выборке метриками качества являются RMSE и MAE(среднеквадратичная ошибка модели и средняя абсолютная соответственно). Обучаются как веса во всех механизмах внимания, так и в матрицах эмбеддингов.

## 0.4 Вычислительный эксперимент

Первоначальной целью моей дипломной работы было не просто разобраться в архитектуре для решения задач рекомендации, но и попробовать переобучить её на задаче рекомендации статей пользователям. В данной главе я подробнее расскажу о выбранных для переобучения данных, их структуре. Опишу, какие сложности возникли при адаптации данных под данную архитектуру, как проделывалась предобработка данных в целом. В конце можно увидеть результаты моих экспериментов. Сразу стоит уточнить, что эксперименты удались и данная архитектура показала неплохие результаты на другой задаче и, вообще говоря, на другого рода данных. Но обо всём по порядку

### 0.4.1 Экспериментальные данные

Для эксперимента был выбран Microsoft News Dataset (данные из сайта Майкрософт новости). Публикацию о создании этого датасета можно посмотреть по ссылке [8].

Набор данных MIND для рекомендаций по новостям был собран из анонимных журналов поведения пользователей (пользовательских логов) на веб-сайте Microsoft News. Случайным образом был отобран 1 миллион пользователей, у которых было не менее 5 кликов по новостям в течение 6 недель с 12 октября по 22 ноября 2019 г. Для защиты конфиденциальности пользователей каждый пользователь хэшируется, что позволяет рассматривать набор читателей просто как набор идентификаторов. Также в оригинальном датасете задействовано порядка 160 тысяч статей.

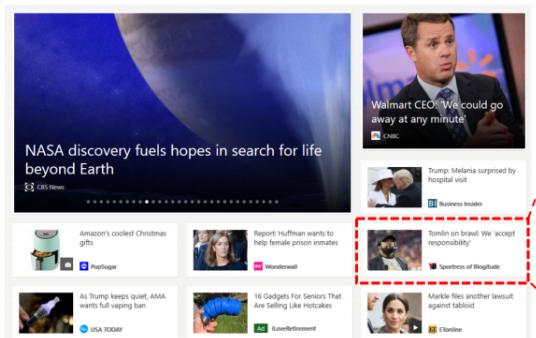
Сразу хочу отметить, что для обучения на локальной машине это слиш-

ком большой объём данных, поэтому специально для этого разработчики датасета Microsoft News сделали уменьшенную версию датасета, которая содержит в себе информацию о предпочтениях 50000 пользователей. Каждый из них отмечал некоторые статьи как понравившиеся или нет. Суммарно задействованных статей оказалось порядка 20000 при последующей обработке данных.

#### 0.4.2 Структура экспериментальных данных

Теперь хотелось бы более подробно описать структуру датасета.

На рисунке 5 можно увидеть, как выглядит новость и какие у неё основные атрибуты: заголовок, категория, краткая выдержка основной информации (abstract) и текст новости. Также необходимо знать, какой пользователь поинтересовался этой новостью и какая история у данного пользователя(что ему понравилось/не понравилось).



(a) An example Microsoft News homepage

<b>Title</b>	Mike Tomlin: Steelers 'accept responsibility' for role in brawl with Browns
<b>Category</b>	Sports
<b>Abstract</b>	Mike Tomlin has admitted that the Pittsburgh Steelers played a role in the brawl with the Cleveland Browns last week, and on Tuesday he accepted responsibility for it on behalf of the organization.
<b>Body</b>	<p>Tomlin opened his weekly news conference by addressing the issue head on.</p> <p>"It was ugly," said Tomlin, who had refused to take any questions about the incident directly after the game, per Brooke Pryor of ESPN. "It was ugly for the game of football. I think all of us that are involved in the game, particularly at this level, ..."</p>

(b) Texts in an example news article

Рис. 5: Пример обычной новости из сайта Майкрософт.

В целом это всё, что нужно знать для поставленной задачи рекомендации статей, и в MIND есть такая информация. Более того, там есть и другие данные, однако я опущу их описание в силу неактуальности для поставленной задачи рекомендации.

## Поведение пользователей

На рисунке 6 можно увидеть, в каком формате хранится информация о пользователях. В целом, это набор впечатлений пользователя от просмотренных статей за одну сессию. У каждого такого набора есть свои атрибуты:

- Идентификатор набора впечатлений(Impression ID).
- Идентификатор пользователя (захэшированное значение в рамках конфиденциальности данных).
- Время начала сессии, в ходе которой был просмотрен ряд статей.
- История просмотра статей данным пользователем.
- Набор впечатлений пользователя о ряде статей.

Column	Content
Impression ID	91
User ID	U397059
Time	11/15/2019 10:22:32 AM
History	N106403 N71977 N97080 N102132 N97212 N121652
Impressions	N129416-0 N26703-1 N120089-1 N53018-0 N89764-0 N91737-0 N29160-0

Рис. 6: Данные о пользовательском поведении

## Структура новости

На рисунке 7 представлена структура новости. Для простоты я решил продемонстрировать неполный список связанных с новостью атрибутов,

дабы не запутать читателя. Можно увидеть, что для каждой новости есть следующий набор атрибутов:

- Идентификатор новости.
- Категория и подкатегория новости.
- Заголовок.
- Краткое содержание(abstract).

Хотел бы обратить внимание на то, что именно по краткому содержанию каждой новости можно генерировать предобученный эмбеддинг каждой новости, что будет подробно описано далее.

Column	Content
News ID	N37378
Category	sports
SubCategory	golf
Title	PGA Tour winners
Abstract	A gallery of recent winners on the PGA Tour.

Рис. 7: Структура новости

### 0.4.3 Исходные данные

Чтобы понять, какую именно информацию необходимо использовать для рекомендации статей, необходимо понимать суть и вид данных для первоначальной задачи рекомендации товаров пользователю. Поэтому для начала я подробно опишу структуру исходного датасета.

Он довольно небольшой, включает в себя знания о 705 пользователях и 1941 товаре из интернет-магазина и хранится в файле формата .pickle. Такой формат прост и удобен в использовании: туда можно загружать всевозможные структуры данных командой `dump` и подгружать данные обратно командой `load`. Теперь о том, что же там внутри:

- История предметов, связанных с пользователем, вместе с соответствующими рейтингами товаров. Данные представлены в виде словаря, где каждому пользователю соответствует список релевантных предметов и их оценка.
- История пользователей, связанных с определённым предметом, с соответствующими рейтингами. По сути, это такой же словарь(как в пункте 1), только составленный для каждого предмета рекомендации. В главе про архитектуру модели я подробно описывал, как эти словари(а именно графовые данные в виде словарей) используются для построения и обучения модели.
- Обучающая и тестовая выборки данных. Один элемент обучения (`sample`) состоит из тройки чисел, задающих номер пользователя, номер предмета и соответствующий рейтинг рекомендации. Далее через матрицу эмбеддингов эти числа преобразуются в соответствующие им эмбеддинги в высокоразмерном пространстве.

- Социальный граф, представляющий собой ещё один словарь, задающий взаимосвязь каждого из пользователей с другими пользователями.

В данных нет валидационного набора данных, только обучающий и тестовый наборы. Да и размер их небольшой: 14091 и 3733 элемента соответственно. Также, все наборы генерируются по сути из одного распределения, так как графовые данные берутся из одного источника. Это не очень хорошо: такая архитектура не является гибкой к новым статьям и пользователям(не добавленным в граф) и не сможет качественно проделывать рекомендацию для них. Однако, такая задача и не является приоритетной, так как зачастую нам заранее известен конкретный граф из пользователей и предметов рекомендации. И для такой задачи архитектура хорошо подходит, обучается и даёт результат. Удастся ли переобучить её на данных новостных статей?

#### 0.4.4 Предобработка экспериментальных данных

Теперь, исходя из знаний об исходных данных, можно смоделировать вид новых данных. Пройдёмся по тем же пунктам, что и в разделе "Исходные данные" для того, чтобы показать процесс моделирования:

- Пробегаясь по логам для каждого пользователя(их было около 156 тысяч для 50 тысяч пользователей соответственно), я отбирал статьи и соответствующие рейтинги статей(0 или 1) в словарь. Итого, для каждого пользователя по его ключу(идентификатору) был получен список релевантных материалов и оценок.
- Чтобы для каждой упомянутой выше статьи получить набор пользователей, каким-либо образом оценивших данную статью, надо было проделать операцию обратного соответствия или инвертирования (новость-пользователи вместо пользователь-новости) через вспомогательные словари.
- Обучающую и тестовую выборки я генерировал из словарей выше. Это довольно удобно: выбираю случайным образом пользователя, случайную релевантную ему статью и соответствующую оценку. Получаю тройку чисел пользователь-новость-оценка.
- С социальным графом возникли наибольшие трудности. Для генерации социального графа я выбрал следующий подход: если у двух разных пользователей количество совместно оценённых статей больше определённого порога, это свидетельствует о схожих предпочтениях пользователей и их надо соединить в социальном графе. Однако, для построения такого графа необходимо  $50000 * 50000 = 2.5 * 10^8$  операций

поиска общих элементов двух списков. По моим расчётом построение такого графа заняло бы 8.5 часов без права на подбор порогового значения: не подошло - начинай сначала.

Решение по построению социального графа было принято следующее: уменьшить объём входных данных до 7 тысяч пользователей и 8741 связанных с ними статей. Ведь, как было упомянуто ранее, набор данных для рекомендации товаров пользователю составлял всего 700 пользователей и 1941 товар. Таким образом время построения социального графа уменьшилось до 1 минуты, и в результате этого количества данных тоже хватило для переобучения модели.

Касательно подбора порогового значения (а именно количества совместно оценённых статей): запустил в цикле построение социального графа для разных пороговых значений и оценил количество связанных пользователей с выбранным исходным. Например, для первого пользователя я получил 308 связанных с ним. Это означает, что 308 пользователей оценили как минимум 10 таких же статей, что и первый. А для третьего лишь 35 пользователей попали в пул "друзей".

## 0.4.5 Тренировка модели

При тренировке модели я тестировал несколько разных гипотез. Во-первых, как влияют различные гиперпараметры на обучение(скорость обучения, размера батча и даже входных эмбеддингов). Ещё более интересным было попробовать предобучить эмбеддинги новостей исходя из имеющейся информации о новостях. Действительно, для каждого идентификатора новости имеем краткое содержание (*abstract*) в виде как правило одного предложения. Было принято решение прогнать такие предложения через модель трансформера "paraphrase-MiniLM-L6-v2" так как скорость работы именно этой архитектуры заметно выше аналогов.

Взаимодействие с такой готовой системой незамысловатое и легко описывается несколькими строками псевдокода:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
sentences = ["A", "B", "C"]
sentence_embeddings = model.encode(sentences)
```

Получаем на выходе матрицу из эмбеддингов предложений - то что нужно! Перед анализом графиков хотелось бы сразу пояснить, почему например обучение с разными скоростями обучения длилось разное количество итераций. То же самое можно видеть и на графиках в разделе "Тестирование модели". Всё дело в том, что в отсутствии валидационного набора данных модель более склонна к переобучению(на обучающей выборке значение функции потерь падает, а на тестовой наоборот растёт). Чтобы побороть эту проблему, применяется метод ранней остановки (early stopping).

Если чуть более подробно, в рассматриваемой архитектуре на обучающей выборке считается значение суммарной на 100 итерациях функции потерь. После каждой эпохи обучения (которая включает в себя примерно пару сотен итераций в зависимости от размера одного батча) модель тестируется с помощью среднеквадратичной (RMSE) и средней абсолютной (MAE) ошибок сразу на тестовых данных. И если модель замечает, что за заранее заданное количество эпох (я использовал значения 5, 7) ошибка на тестовых данных не уменьшается, то обучение останавливается, дабы не переобучиться. Теперь обратимся непосредственно к анализу эксперимен-

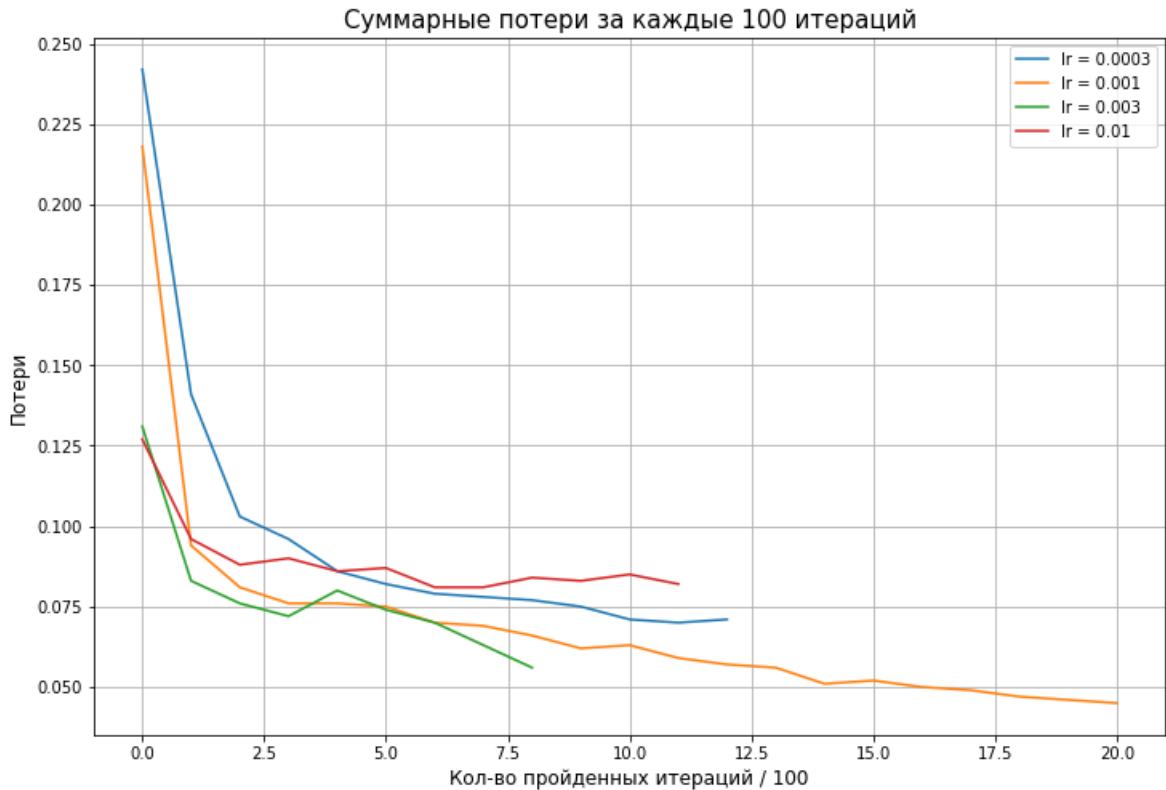


Рис. 8: Суммарные потери за каждые 100 итераций для разных скоростей обучения

тов. Наиболее важные из них я представил в виде готовых графиков. Стоит отметить, что помимо скорости обучения я экспериментировал с размером входных эмбеддингов (набор значений 32, 64, 128), что не дало особых пре-

имуществ или недостатков. Также я менял размер батча (размер обучающей подвыборки, на которой происходит 1 итерация обучения forward-loss-backward) для более быстрых расчётов на GPU в Google Colab. Однако наиболее существенное влияние на функцию потерь при обучении оказывала скорость обучения и на рисунке 8 можно видеть, что наиболее оптимальные параметры скорости равны 0.001 и 0.003.

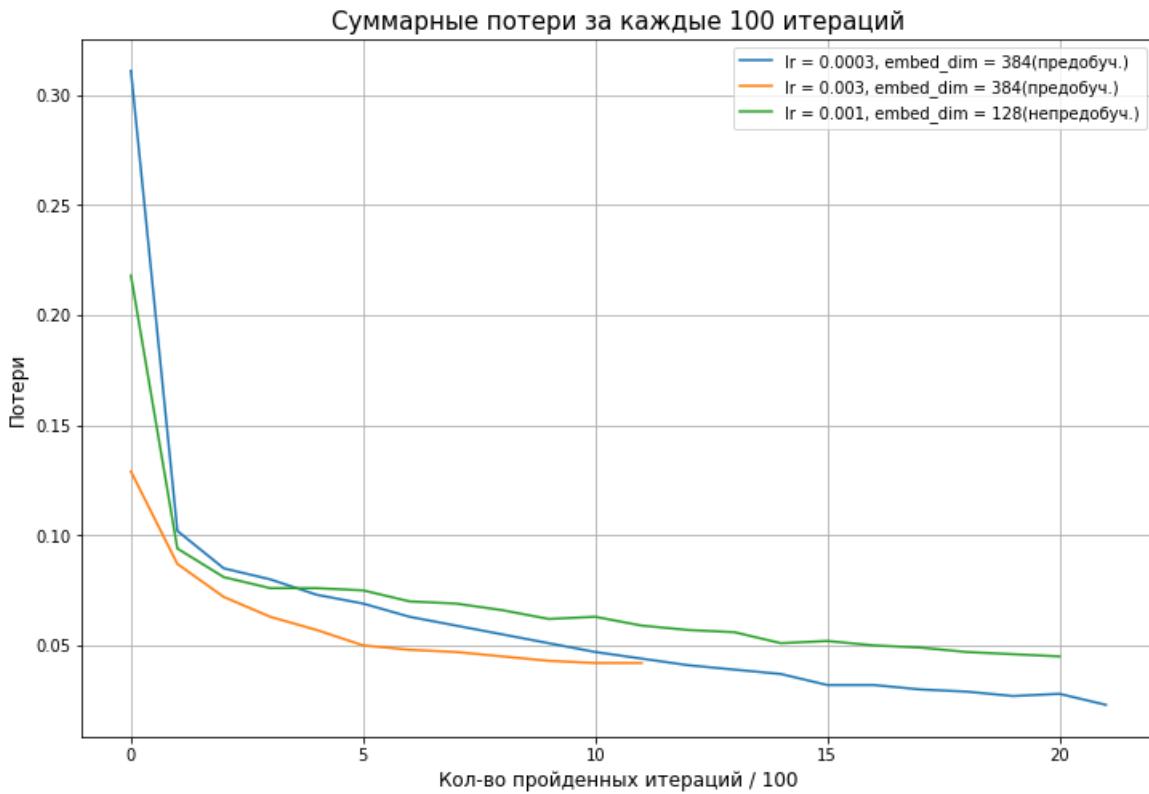


Рис. 9: Суммарные потери за каждые 100 итераций для разных эмбеддингов

На рисунке 9 можно наблюдать, как повлияло внедрение в архитектуру предобученных эмбеддингов новостей(фиксированной размерности 384: это особенность трансформера "paraphrase-MiniLM-L6-v2"). С экспериментальной точки зрения было крайне любопытно узнать, повысится ли качество рекомендательной системы после нововведения. По графику видим, что действительно функция потерь на обучающей выборке немного умень-

шилась по сравнению с лучшим результатом из рисунка 8 для непредобученных эмбеддингов (зелёная ветка). Однако, результат оказался немного хуже ожидаемого, и на тестовых данных чуть ниже мы это увидим.

#### 0.4.6 Тестирование модели

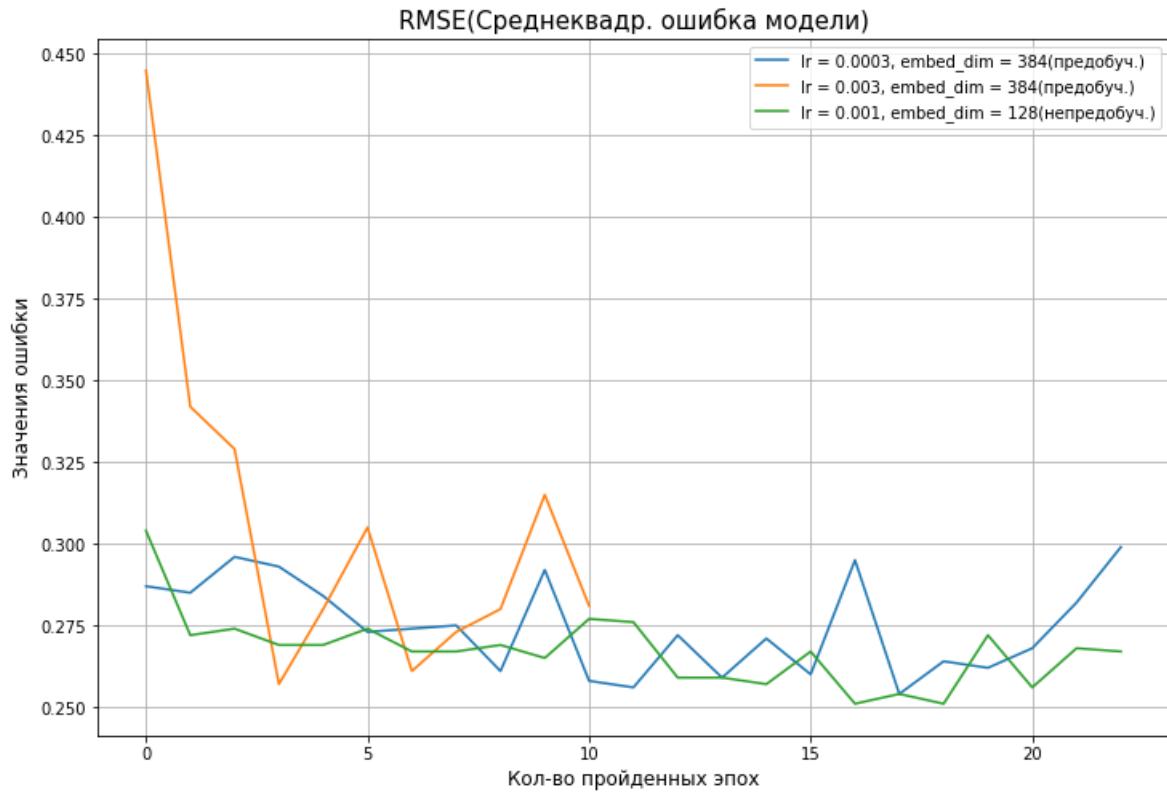


Рис. 10: Среднеквадратичная ошибка модели

Из данных на рисунке 10 можно сделать 2 основных вывода:

- Модель действительно переобучилась под другую задачу рекомендации новостей пользователю. Это видно на зелёной ветке графика, где на тестовой выборке метрика RMSE уменьшилась с 0.3 до примерно 0.25 до начала переобучения уже на обучающей выборке.
- Внедрение преобученных эмбеддингов не дало желаемого результата

на тестовой выборке(синяя и оранжевая ветки на рисунке 10), хотя на обучающей выборке значения функции потерь действительно снизились.

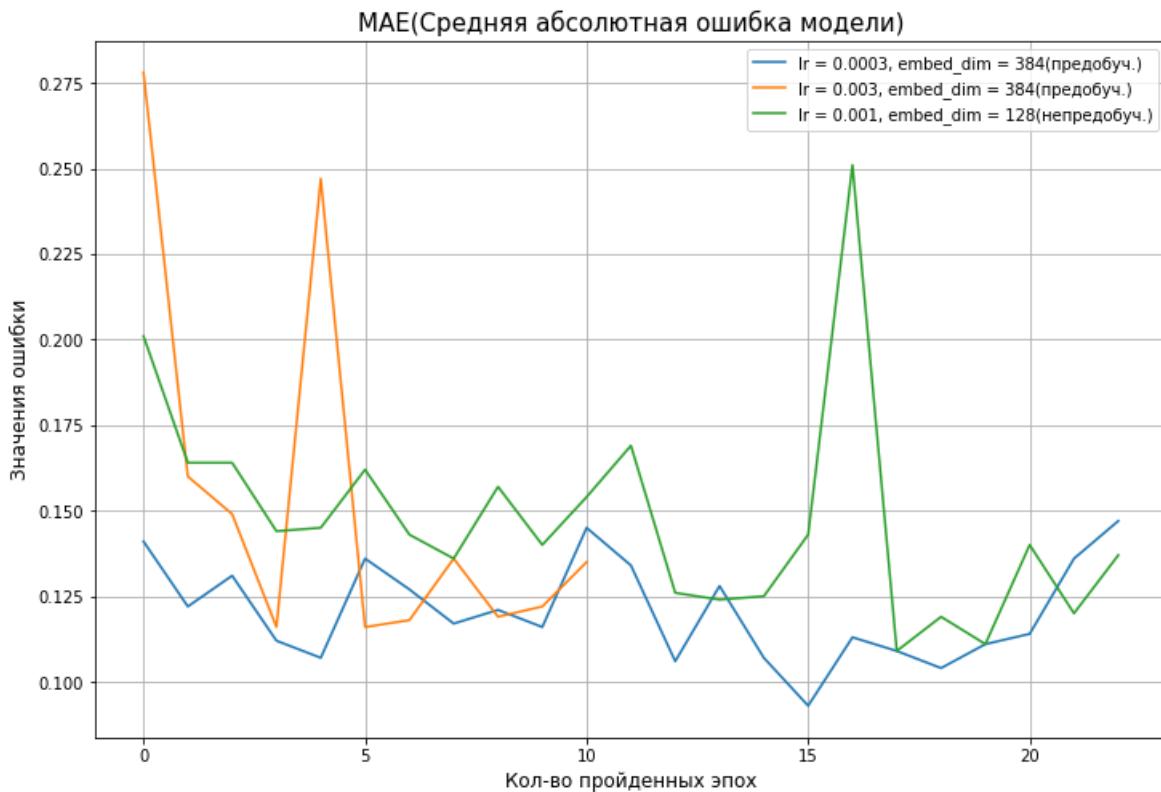


Рис. 11: Средняя абсолютная ошибка модели

Из рисунка 11 однако можно заметить, что ошибки на тестовой выборке после внедрения предобученных эмбеддингов снизились, но незначительно.

## 0.5 Заключение

### 0.5.1 Итоги работы

Данная дипломная работа посвящена исследованию методов графовых нейронных сетей в рекомендательных системах. В работе были рассмотрены несколько подходов к анализу графовых данных для разных задач. Главным фокусом моей работы стала задача предсказания связи (link prediction) между разными объектами (вершинами) графа, так как рекомендательные системы анализируют, связаны ли объекты (например, покупатель и товар) и насколько они релевантны друг другу.

В основной части работы я решил сосредоточиться на одной архитектуре (подробно описанной в разделе "Основная архитектура") и выполнить следующий эксперимент: переобучить исходную модель для рекомендации товаров пользователю на новых данных (Microsoft News Dataset). При удачном исходе выбранная модель должна быть применима для рекомендаций новостных публикаций зарегистрированным пользователям сайта Microsoft News по принципу "вам понравится эта статья / не понравится". В разделе "Вычислительный эксперимент" дано всеобъемлющее описание подготовки экспериментальных данных, а также результаты обучения и тестирования выбранной архитектуры на подготовленных данных.

Главным итогом работы является то, что выбранная архитектура действительно смогла переобучиться под другую задачу рекомендации с относительно неплохими результатами по метрикам RMSE, MAE.

Также был проведён эксперимент по улучшению качества рекомендательной системы с помощью внедрения предобученных эмбеддингов новостных публикаций в архитектуру. Эксперимент не удался полностью, хотя неболь-

шое улучшение всё же можно наблюдать, о чём я более подробно написал в пункте "Тестирование модели".

### 0.5.2 Дальнейшие исследования

Выбранная мной тема дипломной работы оказалась довольно объёмной, но в то же время интересной. Поэтому хотелось бы вкратце рассказать о том, какие эксперименты ещё можно провести для улучшения качества рекомендаций выбранной архитектуры. Среди них:

- Внедрить предобученные эмбеддинги пользователей в текущую версию архитектуры. Сгенерировать данные эмбеддинги исходя из новостных предпочтений пользователя.
- Применить методы тематического моделирования для генерации более информативных эмбеддингов новостных статей.

# Литература

- [1] Graph Neural Networks for Social Recommendation / Wenqi Fan, Y. Ma, Qing Li et al. // The World Wide Web Conference. — 2019.
- [2] Graph Neural Networks for Social Recommendation / Wenqi Fan, Yao Ma, Qing Li et al. // The World Wide Web Conference / ACM. — 2019. — P. 417–426.
- [3] Newman M. Clustering and preferential attachment in growing networks. // Physical review. E, Statistical, nonlinear, and soft matter physics. — 2001. — Vol. 64 2 Pt 2. — P. 025102.
- [4] Barabási, Albert. Emergence of scaling in random networks // Science. — 1999. — Vol. 286 5439. — P. 509–12.
- [5] Similarity measures in scientometric research: The Jaccard index versus Salton’s cosine formula / L. Hamers, Yves Hemeryck, Guido Herweyers et al. // Inf. Process. Manag. — 1989. — Vol. 25. — P. 315–318.
- [6] Zhang Muhan, Chen Yixin. Link prediction based on graph neural networks // Advances in Neural Information Processing Systems. — 2018. — P. 5165–5175.

- [7] Kipf Thomas, Welling M. Semi-Supervised Classification with Graph Convolutional Networks // ArXiv. — 2017. — Vol. abs/1609.02907.
- [8] MIND: A Large-scale Dataset for News Recommendation / Fangzhao Wu, Ying Qiao, Jiun-Hung Chen et al. // ACL. — 2020.
- [9] A capsule network-based embedding model for knowledge graph completion and search personalization / Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen et al. // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). — 2019. — P. 2180–2189.
- [10] Sara Sabour Nicholas Frosst Geoffrey E Hinton. Dynamic Routing Between Capsules. — 2017.