

Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа Радиотехники и Компьютерных Технологий
Кафедра банковских информационных технологий

Направление подготовки / специальность: 03.04.01 Прикладные математика и физика
(магистратура)

Направленность (профиль) подготовки: Математические и информационные технологии

ПРИМЕНЕНИЕ NLP МЕТОДОВ ГЛУБОКОГО ОБУЧЕНИЯ В КОНТЕКСТНЫХ РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМАХ

(магистерская диссертация)

Студент:
Тышко Алексей Андреевич

(подпись студента)

Научный руководитель:
Кантор Виктор Викторович,

(подпись научного руководителя)

Консультант (при наличии):

(подпись консультанта)

Москва 2020

Содержание

1 Введение	2
2 Методология работы	12
2.1 Данные	12
2.2 Методология	13
2.3 Предобработка данных	21
3 Реализация предложенных моделей и подходов	25
4 Результаты	32
5 Выводы	36

1 Введение

За последние несколько десятилетий, с ростом и развитием таких сервисов как Amazon, YouTube, Netflix и других им подобных, рекомендательные системы приобрели большую значимость в нашей жизни. От сферы электронной коммерции (все финансовые и торговые транзакции, осуществляемые при помощи компьютерных сетей, и бизнес-процессы, связанные с проведением таких транзакций) и до сферы интернет-рекламы рекомендательные системы помогают находить и предлагать пользователям релевантный контент или продукт.

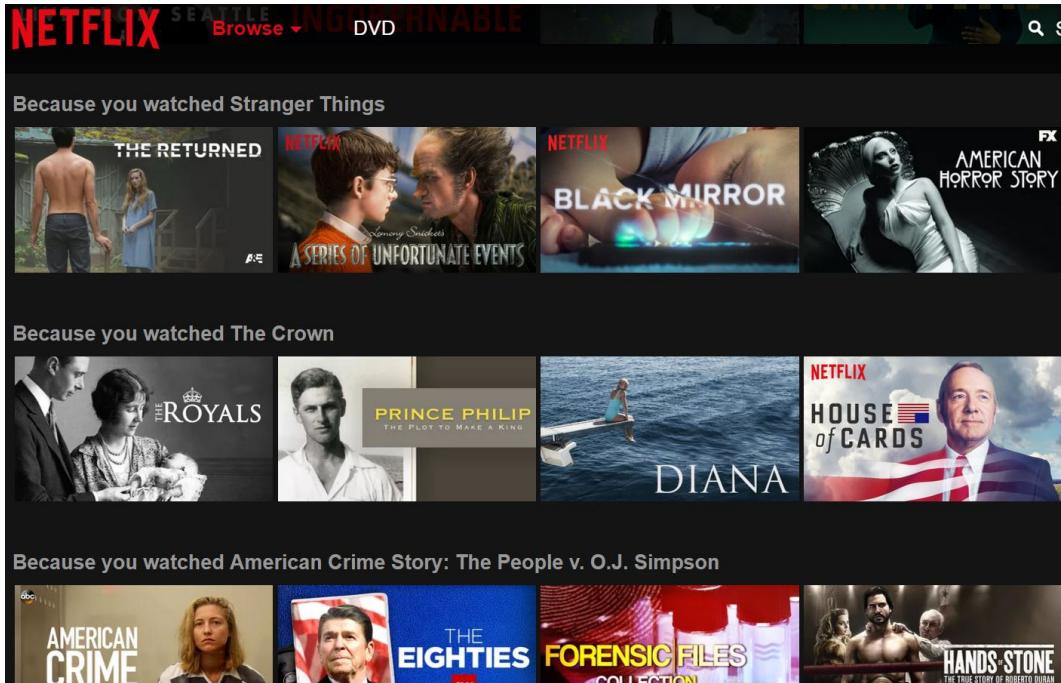


Рис. 1: Пример рекомендации фильмов на сайте Netflix [1].

Рекомендательные системы очень важны во многих индустриях, поскольку они могут приносить огромные доходы в случае, когда они эффективны, а также могут послужить хорошим средством для борьбы с конкурирующими предприятиями. В качестве примера стоит упомянуть соревнование «the Netflix Prize» организованное компанией Netflix, в котором целью было создать такую рекомендательную систему, которая превзошла бы собственную рекомендательную систему Netflix'a, а в качестве приза был один миллион долларов.

В общих чертах, рекомендательные системы это алгоритмы (комплексы алгоритмов) которые пытаются предсказать какие объекты (товары, продукты, фильмы, музыка, книги, новости) будут интересны данному пользователю, имея некую информацию о самом пользователе.

Рекомендательные системы могут быть построены с использованием многих различных парадигм, от простых (основанных только на оценках других продуктов того же пользователя) до чрезвычайно сложных. В таких сложных системах используются различные типы и источники данных (например

можно использовать изображения товаров, которые предлагается рекомендовать пользователю) а также различные техники машинного и глубокого обучения. Таким образом современные рекомендательные системы вовлекаются в мир Искусственного Интеллекта и Больших Данных, весьма популярные и активно развивающиеся области нашего времени.

Существует два основных подхода к классическим рекомендательным системам. В зависимости от преследуемых целей, аудитории пользователей, платформы и самих объектов, которые предлагается рекомендовать, эти подходы можно использовать по отдельности, хотя зачастую лучший результат достигается засчет их комбинирования.

Первый подход называется Коллаборативная фильтрация [2] (здесь и далее в разделе Введение, если явно не указана ссылка, все новые термины заимствованы из данной статьи). Он преимущественно делает рекомендации на основе данных или действий других пользователей, отличных от пользователя, которому мы хотим сделать рекомендацию.

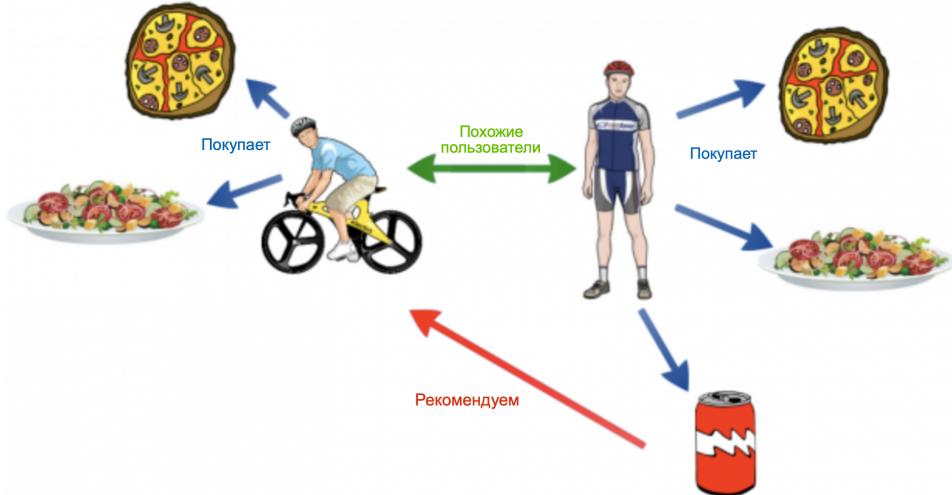


Рис. 2: Двум похожим пользователям рекомендуются одинаковые товары [3].

Существуют две вариации рекомендательных систем такого типа:

- **Рекомендации на основе групп пользователей:** этот способ предполагает создание групп пользователей на основании сравнения их активности и действий, и рекомендуются товары, популярные среди других членов такой группы. Такой способ полезен на платформах с обширной и разнообразной аудиторией, он позволяет делать быстрые рекомендации пользователям, по которым достаточно мало информации.
- **Рекомендательная система на основе ассоциативного правила:** кратко этот способ можно описать вопросом «Какие товары чаще всего встречаются вместе?». То есть, перефразировав, «Какие товары вместе с данным берут пользователи, похожие на пользователя, для которого

делается рекомендация?». В данном подходе рекомендуются товары в режиме реального времени, на основе товаров которые пользователь просмотрел (или добавил в корзину) в данный момент (в данную сессию).

Второй подход называется контент-ориентированная фильтрация. Он основывается на накапливании истории покупок или сессий пользователя. Как правило, точность данного подхода увеличивается по мере увеличения истории действий пользователя.

Контент-ориентированная фильтрация

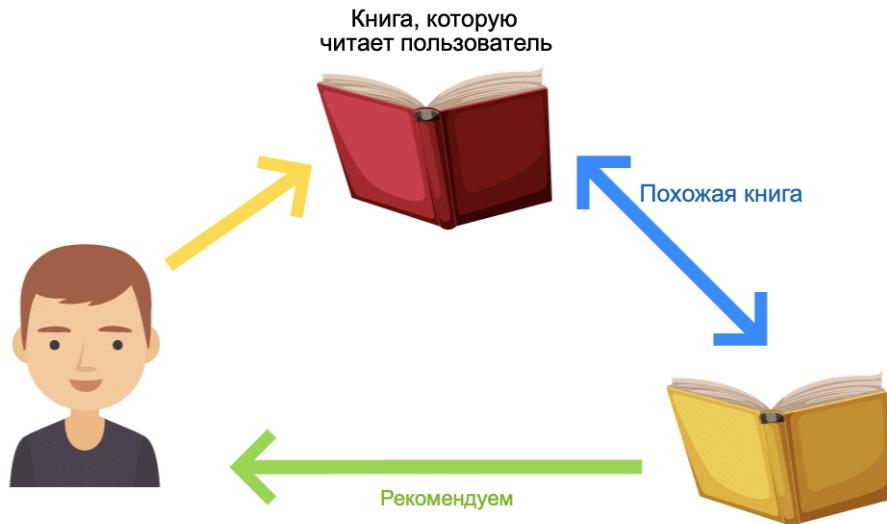


Рис. 3: Одному пользователю рекомендуется книга, схожая с той, которую он читал раньше [3].

Частные случаи контент-ориентированной рекомендательной системы включают:

- **Рекомендательная система на основе сходства контента:** наиболее базовый подход контент-ориентированных рекомендательных систем, построенный на основе поиска похожего контента относительно его метаданных (признаков). Этот подход стоит применять в случаях, когда имеется большой объем метаданных, но с количеством признаков меньшим, чем количество всех продуктов.
- **Рекомендательная система на основе моделирования скрытых факторов:** суть этого подхода заключается в выявлении индивидуальных предпочтений пользователей исходя из предположения о том, что предыдущие покупки и взаимодействия с товарами характеризуют определенные вкусы покупателей. Если более формально дать определение этому подходу, то он заключается в том, чтобы по имеющимся данным взаимодействия пользователей с товарами определить некие частные, ранее неизвестные и независимые признаки самих пользователей и товаров.

- **Рекомендательная система на основе тематического моделирования:** это разновидность моделирования скрытых факторов, где вместо того, чтобы исследовать огромное количество взаимодействий пользователей с товарами анализируют неструктурированные тексты для выявления неких абстрактных «тем интересов». Этот подход имеет смысл применять в случае наличия большого количества текстовой информации (например такой, как новостные статьи).
- **Рекомендательная система на основе продвижения популярного контента:** этот подход включает в себя отбор продукта для рекомендаций, основанный на его внутренних характеристиках (цена, популярность), которые могут привлечь достаточно широкую аудиторию. Этот способ также учитывает новизну или возраст контента и таким образом может рекомендовать товары из трендов. Такую разновидность рекомендательных систем часто используют, когда имеется много нового контента.

Базовые рекомендательные системы существуют уже довольно давно, и хотя они и усложняются и совершенствуются, несмотря на это появляются все новые и новые тренды развития подобных современных систем:

- **Контекстные рекомендательные системы** представляют новую область исследований и экспериментов, направленных на определение более подходящего контента для рекомендаций пользователю в определенный момент времени. Такие рекомендательные системы учитывают, например, находится ли пользователь дома или нет, использует ли он большой экран устройства или маленький, какое время суток сейчас, время года. Агрегируя всю эту информацию такая система способна давать более точные и релевантные рекомендации.
- **Глубокое обучение** в задачах рекомендаций уже успешно используется некоторыми компаниями гигантами, такими как YouTube или Spotify. Но по мере стремительного роста объема накапливаемых данных, большие корпорации сталкиваются с проблемой масштабирования данных и глубокое обучение неизбежно становится методологией не только для решения задач рекомендательных систем, но и для большинства других «задач обучения».
- **Решение проблемы холодного старта** также является одним из тех направлений, в которое сейчас современные исследователи вкладывают много сил. Проблема заключается в рекомендации товаров, по которым мало или практически не накоплено никакой информации. Решение данной задачи очень важно для компаний с большим объемом контента, оно способно определить какие товары стоит продвигать к рекомендациям еще до того, как их запустят в оборот.

Рекомендательные системы должны быть эффективным способом показывать пользователям контент, который они сами с большой вероятностью никогда бы не нашли. Это в свою очередь должно способствовать достижению крупных бизнес-целей, таких как увеличение продаж, доходов от рекламы или привлечение новых пользователей. Построение сложной системы, требующей опытного персонала

и постоянного обслуживания, когда работает достаточно простое и дешевое решение, окажется просто пустой тратой денег, поэтому рекомендательная система должна действительно играть важную роль в бизнес-процессах. Также такая система должна быть гибкой, то есть адаптироваться и развиваться соответственно пользовательским предпочтениям. Поэтому разработка рекомендательной системы это всегда постоянный контроль за развитием поведения пользователей, контроль за тем, что работает, а что нет, продумывание дополнительных источников данных, все это ради усовершенствования имеющейся системы, которое поможет делать лучшие рекомендации.

В последние годы количество научных публикаций по применению глубокого обучения в задачах рекомендательных систем увеличилось в геометрической прогрессии. Такие системы имеют 4 основных преимущества над классическими Коллаборативной и Контент-Ориентированной фильтрациями.

Глубокое обучение может моделировать нелинейные взаимодействия в данных с помощью нелинейных активаций, таких как ReLU, Sigmoid, Tanh [4]. Эта способность позволяет выявлять сложные и запутанные взаимодействия пользователя и продукта. Традиционные методы, о которых говорилось выше, по сути являются способами разложения и понижения размерности матриц, причем при их использовании выдвигается предположение о линейной зависимости в данных, что в свою очередь значительно упрощает и ограничивает результативность модели. Хорошо известно, что нейронные сети способны аппроксимировать любую непрерывную функцию с высокой точностью путем варьирования ее коэффициентов. Это свойство позволяет иметь дело со сложными рисунками взаимодействия в данных и более точно отражать предпочтения пользователей.

Глубокое обучение способно обобщать зависимости в исходных данных а также создавать их весьма полезные признаковые представления. Как правило, в современных приложениях содержится очень много описательной информации о пользователях и товарах. Использование этой информации позволяет нам усилить понимание взаимосвязей между самими товарами и пользователями, что может привести к улучшению рекомендаций. Таким образом имеет смысл использовать глубокие нейронные сети для получения признаковых представлений в рекомендательных моделях. Все это может сократить труд, затрачиваемый на ручное создание признаков для модели, а также добавлять в модель дополнительную разнородную информацию, такую как текст, изображения, аудио или даже видео.

Глубокое обучение является мощным инструментом для задач с моделированием последовательностей. В таких задачах, как машинный перевод, обработка естественного языка, распознавание речи и так далее, рекуррентные и сверточные нейронные сети играют определяющую роль. Они широко применяются в обработке и анализе последовательных структур данных. Моделирование последовательностей является важной темой для анализа временной динамики поведения пользователей и эволюции товаров. Например самое частое применение таких нейронных сетей встречается в задачах предсказания следующего товара в корзине или в рекомендациях, основанных на сессии пользователя. Таким образом глубокие нейронные сети становятся идеальным решением для решения

задач в которых фигурируют последовательности.

Глубокое обучение обладает высокой гибкостью. В настоящее время существует множество популярных фреймворков глубокого обучения. Эти фреймворки имеют модульную структуру и активно поддерживаются сообществом и профессионалами. Хорошая модульность делает разработку и проектирование намного более эффективными. Например, можно легко комбинировать различные архитектуры нейронных сетей, чтобы получать мощные гибридные модели или заменять один модуль другим. Таким образом, это позволяет легко строить композитные модели для рекомендаций с возможностью одновременного учета различных признаков и факторов.

Большинство новейших методологий, техник и подходов глубокого обучения нашли себе применение в задачах рекомендательных систем:

- Рекомендации с использованием многослойного перцептрана.

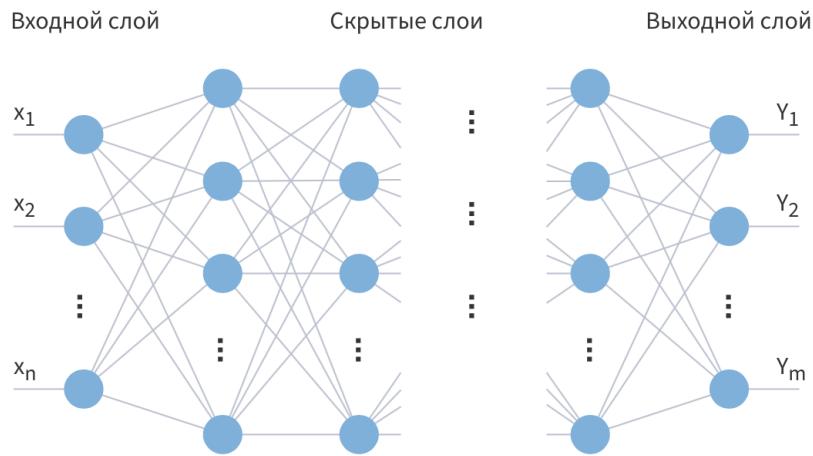


Рис. 4: Схематичная архитектура многослойного перцептрана [5].

Многослойный перцептрон [6] представляет собой несколько последовательно идущих друг за другом полносвязных слоев с нелинейными трансформациями, которые иерархически выучивают скрытые признаковые описания. Это довольно простая модель, которая, однако, может весьма успешно извлекать скрытые признаки (факторы) пользователей и товаров, а также хорошо моделирует их двухстороннее взаимодействие.

- Рекомендации с использованием автокодировщика.

Автокодировщик [7] - это модель, которая в процессе обучения изменяет входные данные, а затем пытается восстановить их в выходном слое. В общем случае архитектура подобной сети такова, что вначале она сжимает исходные данные (понижает размерность входных векторов), а потом расширяет до исходного размера. Слой с самой маленькой размерностью («узкое горлышко») используется в качестве векторного представления признакового описания входных данных. Таким

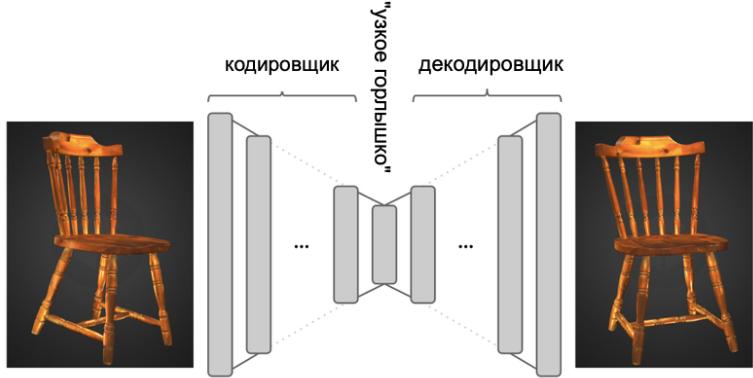


Рис. 5: Схематичное изображение работы автокодировщика [3].

образом с помощью автокодировщика можно решать задачу определения скрытых признаков (факторов) пользователей и товаров, а также задачу заполнения пропусков в матрице рейтингов пользователь/товар.

- Рекомендации с использованием сверточных нейронных сетей.

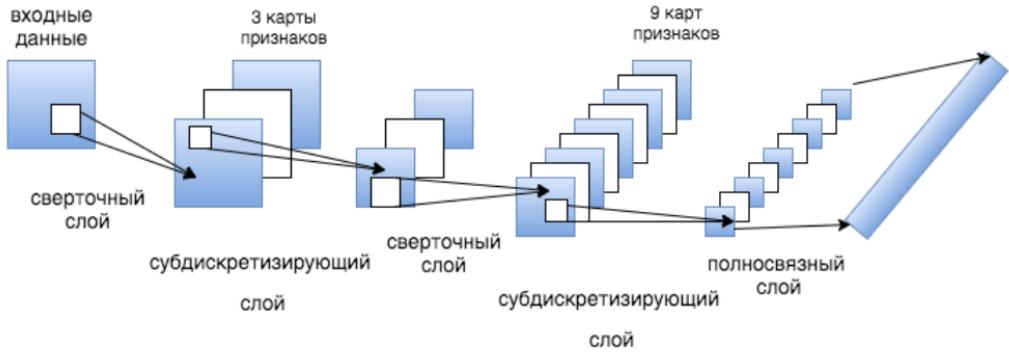


Рис. 6: Схематичная архитектура сверточной нейронной сети [8].

Сверточная нейронная сеть [9] - это сеть, состоящая из сверточного слоя, слоя активации и слоя субдискретизации, изначально разработанная для задач распознавания образов и обработки изображений. Такие сети могут извлекать признаковое описание из картинок, которое в дальнейшем можно добавлять к имеющимся признакам товаров. Также сверточные сети используют в задачах колаборативной фильтрации для понижения размерности матрицы взаимодействий пользователей.

тель/товар.

- Рекомендации с использованием рекуррентных нейронных сетей.

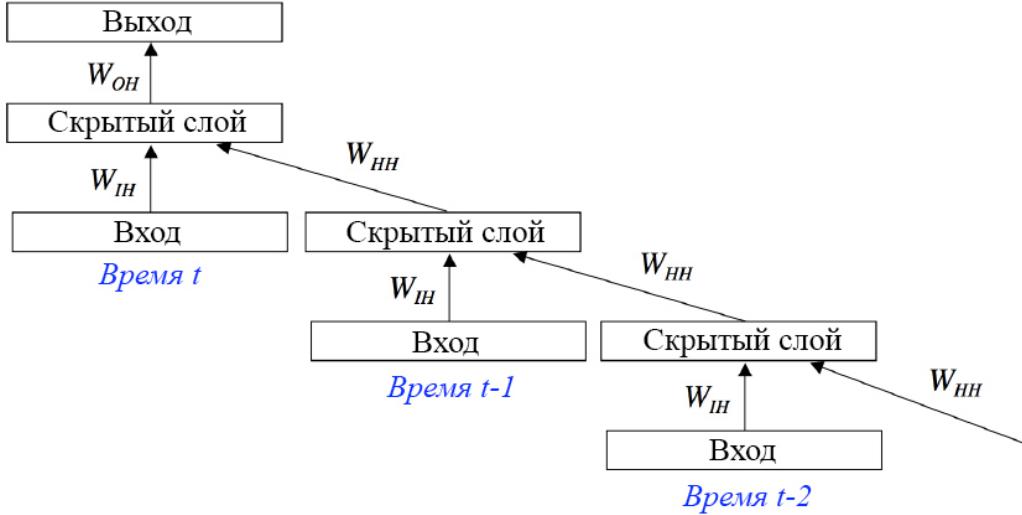


Рис. 7: Схематичная архитектура развернутой рекуррентной сети [10].

Рекуррентные нейронные сети [11] - это сети с циклами, которые подходят для обработки серии событий во времени или последовательные пространственные цепочки. Такие сети используются для моделирования рисунков поведения пользователей в задачах рекомендаций, основанных на сессии пользователя, а также для извлечения скрытых признаков этих сессий.

- Рекомендации с использованием Механизма Внимания.

Механизм Внимания [12] - это подход в машинном обучении, заключающийся в выделении наиболее важных частей во входных данных и в фильтрации незначащих признаков для снижения побочных эффектов в зашумленных данных. В задачах рекомендательных систем используют данный подход для выявления наиболее репрезентативных товаров.

Глубокое обучение становится все более популярным во многих областях, включая обработку естественного языка, в частности два последних описанных выше подхода играют большую роль в этой области и сейчас методы и приемы из нее активно применяются исследователями в задачах рекомендательных систем.

В работе [13] исследователи предложили новый метод для колаборативной фильтрации на основе признаков товаров. Данный метод базируется на подходе под названием скип-грамм с отрицательной выборкой, который был предложен Миколовым и группой исследователей [14]. Этот подход направлен на поиск векторного представления слова и учитывает взаимосвязи между соседними словами, находящимися с целевым словом в одном предложении. В этой работе использовались два различных набора

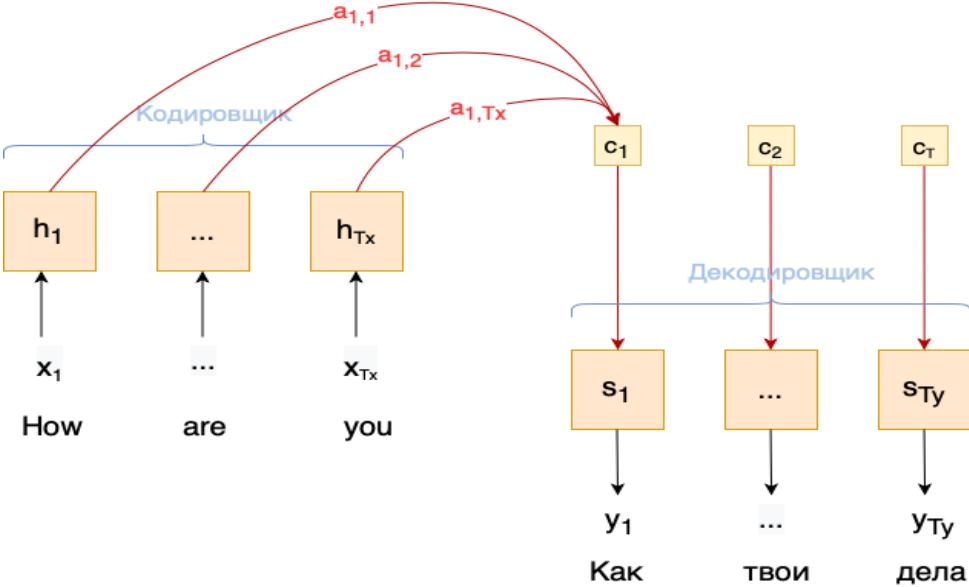


Рис. 8: Схематичное изображение работы Механизма Внимания.

данных из Microsoft Xbox Music и Microsoft Store. Для получения скрытых признаков товаров, исследователи обучали алгоритм скрип-грамм на группах взаимодействий пользователя с исполнителем в случае первого набора данных, а для второго они обучали тот же алгоритм на товарах, которые пользователи добавляли в свою корзину. В случае первого набора данных имелась информация о взаимодействиях пользователя с товарами, то есть матрица взаимодействия пользователя с товаром, в то время как для второго набора данных при проведении эксперимента никакой информации о пользователе, который собрал ту или иную корзину товаров не использовалось. Таким образом для первого набора данных исследователи сравнили свой новый подход понижения размерности матрицы, и стандартный подход с использованием Сингулярного разложения. Затем они провели кластеризацию данных по жанрам музыки для первого набора данных, и для обоих наборов делали рекомендации товаров таким образом, что для одного товара рекомендовались товары, близкие к нему с точки зрения алгоритма K Ближайших соседей [15]. В результате модель, основанная на подходе скрип-граммы с отрицательной выборкой показала лучшие результаты в сравнении с классическим сингулярным разложением. Возможным недостатком данной работы является слабая базовая модель, с которой сравнивалась целевая модель.

В работе [16] исследователи провели попытки использовать рекуррентные сети для рекомендаций основанных на сессии пользователя, в частности они использовали модификацию обычных рекуррентных сетей под названием Управляемый рекуррентный блок [17]. Использовались данные из соревнования RecSys Challenge 2015 года и с видео-платформы YouTube-like OTT. Это два различных набора данных, первый из которых состоял из последовательностей кликов пользователя на сайте товаров, второй набор состоял из последовательностей событий просмотра видео пользователем. Для обучения

данные последовательности событий были разбиты на независимые сессии и решалась задача предсказания следующего события (клик на сайте) в сессии. Таким образом для рекомендации товаров необходимо было, чтобы пользователь совершил первый клик на сайте. Для сравнения в качестве базовых моделей были использованы такие подходы, как: рекомендация самого популярного товара, рекомендация самого популярного товара внутри текущей сессии, а также использовались рекомендации похожих с текущим (который пользователь просматривает в данный момент) товаром других товаров с точки зрения алгоритма K Елизайших соседей [15]. В качестве целевой метрики использовалась полнота для двадцати рекомендуемых товаров. В результате данный нейросетевой подход улучшил качество примерно на 20-30%, что дает веское основание использовать его в остальных задачах рекомендаций, основанных на сессии пользователя. Недостатком данного подхода может быть проблема холодного старта в случае нового продукта, когда нет никакой накопленной информации о нем.

В статье [18] исследователи использовали модель Word2Vec, предложенную Миколовым и группой исследователей [19] для получения векторного представления контекста пользователя в задаче рекомендательной системы, основанной на сессиях. Исследователи использовали данные с сайта по продаже одежды с ее небольшими текстовыми описаниями (размер, бренд, модель и тд.). Каждая сессия пользователя представляла собой некую последовательность описаний товаров, на которые он кликал, на таких последовательностях исследователи обучили модель Word2Vec и затем кластеризовали полученные векторы товаров. В итоге они получили кластеры стилей одежды, вектора этих кластеров это по сути и были вектора контекста сессии пользователя. Для каждой сессии пользователя получался ее вектор путем усреднения векторов описаний товаров, на которые кликал пользователь во время сессии, и для рекомендации брался самый близкий к этому вектору сессии вектор товара. Таким образом в данной работе был предложен способ получения контекста сессии пользователя. Минусом данной работы является то, что исследователи не провели сравнение работы их метода с классической базовой моделью, что не дает возможности оценить ценность данного подхода.

В данной работе сравниваются вышеописанные подходы с классическим в задачах рекомендаций, основанных на сессиях, в рамках имеющихся данных взаимодействия пользователя с товаром, а также проводится исследование на возможность применения новейших, ранее не применявшимися, state-of-the-art моделей из области обработки естественного языка в контекстных рекомендательных системах для улучшения базовой модели в случае краткой описательной текстовой информации товаров, чем обуславливается ее актуальность.

2 Методология работы

2.1 Данные

В этой работе использовались данные, предоставленные компанией Instacart для соревнования на сайте Kaggle под названием «Instacart Market Basket Analysis» [20]. Данные представляют собой набор связанных между собой файлов, описывающих заказы клиентов в течение некоторого времени. Это анонимный набор данных, содержащий в себе более чем 3 миллиона продуктовых заказов от более чем 200000 пользователей Instacart. Для каждого пользователя предоставлено от четырех до ста заказов, с последовательностями продуктов, приобретенных в каждом заказе. Также имеется информация о неделе и часе дня, когда был сделан заказ, а также число дней, прошедших между двумя последовательными заказами для каждого пользователя.

products				aisles	
product_id	product_name	aisle_id	department_id	aisle_id	aisle
1	Chocolate Sandwich Cookies	61	19	1	prepared soups salads
2	All-Seasons Salt	104	13	2	specialty cheeses

order_products__train				departments	
order_id	product_id	add_to_cart_order	reordered	department_id	department
1	49302	1	1	1	frozen
1	11109	2	1	2	other

orders						
order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
2539329	1	prior	1	2		8
2398795	1	prior	2	3		7 15.0

Рис. 9: Разбитые по файлам данные в формате .csv. В файле products.csv находятся уникальные идентификаторы продуктов, их текстовые названия, и идентификаторы двух категорий, к которым они относятся. В файле aisles.csv содержатся уникальные идентификаторы и текстовые названия первой категории. В файле departments.csv содержатся уникальные идентификаторы и текстовые названия второй категории. В файле order_products_train.csv содержатся уникальные идентификаторы заказа и упорядоченные идентификаторы продуктов, купленных в данный заказ, а также содержится метка для каждого продукта, был ли он куплен в предыдущие заказы конкретным покупателем, то есть является ли заказ этого продукта первым в истории покупателя или нет. В файле orders.csv содержатся уникальные идентификаторы заказов, соответствующие уникальным идентификаторам пользователей, которые данные заказы совершили, еще содержатся признаки заказа (день недели, час дня и количество дней, прошедших с предыдущего заказа), а также содержится метка, является ли этот заказ первым у данного покупателя или нет.

2.2 Методология

Суть данной работы заключается в том, чтобы опробовать различные техники и приемы из области обработки естественного языка в задачах контекстных рекомендательных систем. В работе применяются техники и парадигмы, описанные в том числе и в разделе Введение, и сравниваются с базовой моделью, для которой генерируются отдельные признаки из набора данных, описанных в подразделе Данные.

В качестве базового алгоритма использовалась модель градиентного бустинга над деревьями, которая разрабатывалась Лео Брейманом и была усовершенствована Джеромом Гарольдом Фридманом [21]. Это алгоритм машинного обучения, работающий для задач регрессии и классификации и представляющий собой ансамбль зачастую простых моделей, которые последовательно в процессе обучения пытаются минимизировать ошибки предыдущих моделей. Этот алгоритм работает по принципу градиентного спуска, где в качестве пространства параметров, в которых происходит оптимизация, берется пространство функций (простых алгоритмов). Классический алгоритм градиентного бустинга решает общую задачу обучения с учителем, то есть задачу, в которой имеется набор признаков объекта и некая целевая метка для этого объекта, предсказывать которую и учится алгоритм. На вход алгоритма подается несколько составляющих:

- Набор $\{(x_i, y_i)\}_{i=1,\dots,n}$, где x_i это признаки объектов, а y_i это метки;
- Число итераций M ;
- Выбор функции потерь $L(y, f)$ с выписаным градиентом, где y это реальные метки объектов, а $f(x)$ это предсказанные значения меток алгоритмом;
- Выбор семейства функций базовых алгоритмов $h(x, \theta)$, с процедурой их обучения, где x это признаки объектов, а θ параметры алгоритмов;
- Дополнительные гиперпараметры $h(x, \theta)$, например глубина дерева решений;

В качестве начального приближения $f_0(x)$ (первый алгоритм в ансамбле) берется обычно константный алгоритм γ , который возвращает постоянное число ρ , которое подбирается простым линейным методом относительно исходной функции потерь (а не градиента). Таким образом алгоритм можно описать в виде:

- 1. Инициализировать алгоритм начальным значением $\hat{f}(x) = \hat{f}_0$, $\hat{f}_0 = \gamma$, $\gamma \in R$

$$\hat{f}_0 = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma) \quad (1)$$

- 2. Для каждой итерации $t = 1, \dots, M$ повторять:

- 1. Посчитать псевдо-остатки r_t

$$r_{it} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]_{f(x)=\hat{f}(x)}, \quad i = 1, \dots, n \quad (2)$$

- 2. Построить новый базовый алгоритм $h_t(x)$ как регрессию на псевдо-остатках $\{(x_i, r_{it})\}_{i=1, \dots, n}$
- 3. Найти оптимальный коэффициент ρ_t при $h_t(x)$ относительно исходной функции потерь

$$\rho_t = \arg \min_{\rho} \sum_{i=1}^n L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta)) \quad (3)$$

- 4. Сохранить

$$\hat{f}_t(x) = \rho_t \cdot h_t(x) \quad (4)$$

- 5. Обновить текущее приближение $\hat{f}(x)$

$$\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}_t(x) = \sum_{i=0}^t \hat{f}_i(x) \quad (5)$$

- 3. Скомпоновать итоговую модель бустинга $\hat{f}(x)$

$$\hat{f}(x) = \sum_{i=0}^M \hat{f}_i(x) \quad (6)$$

Для обучения базового алгоритма составлялась выборка с искусственно созданными признаками пользователя, продукта и контекста (то есть признаки заказа: день недели, час и тд.) и ставилась метка 1, если данный пользователь купил данный товар при данном контексте, и ставилась метка 0, если данный пользователь не покупал данный товар при данном контексте. Таким образом решалась задача классификации на два класса, другими словами решалась задача минимизации логистической функции потерь:

$$-\frac{1}{n} \sum_{i=1}^n (y_i \log a + (1 - y_i) \log(1 - a)) \rightarrow \min_a \quad (7)$$

где n – это общее количество объектов, y_i – это метка (класс 1 или 0) i -го объекта, a – это вероятности класса объекта, которые предсказывает алгоритм.

В нашей задаче классификации вероятности считаются с помощью функции сигмоиды:

$$a = sigmoid(y_{pred}) \equiv \frac{1}{1 + e^{-y_{pred}}} \quad (8)$$

где y_{pred} – это некое число от $-\infty$ до $+\infty$, которое выдает наш алгоритм, поэтому и применяется сигмойда, чтобы это число перевести в вероятность, то есть в число от 0 до 1.

Для построения рекомендаций для каждого нового заказа (для которого мы хотим сделать рекомендации пользователю по продуктам), брались продукты с наибольшей вероятностью с точки зрения обученного алгоритма.

В работе использовалось множество подходов и моделей из области обработки естественного языка. Первый подход можно охарактеризовать как классификацию последовательностей продуктов в заказе. Составлялась такая выборка: для каждого пользователя и для каждого его заказа бралась последовательность продуктов, купленная тем же пользователем в предыдущий заказ, затем для каждого продукта в новом заказе составлялась новая последовательность, бралась последовательность продуктов из предыдущего заказа и ей в конец ставился продукт, купленный в новый заказ и этой последовательности присваивалась метка (класс) 1, а если какой-либо продукт не был куплен в новый заказ, то он также добавлялся в конец последовательности, но тогда ей присваивалась метка 0. Таким образом составлялась абсолютно такая же обучающая выборка, как и для обучения базовой модели, но вместо искусственно созданных признаков брались такие последовательности. В качестве алгоритма, который предстояло обучать брался алгоритм под названием Долгая краткосрочная память (LSTM) [22], предложенная в 1997 году Зеппом Хохрайтером и Юргеном Шмидхубером. Этот алгоритм отличается от обычной рекуррентной сети тем, что обладает вентилями забывания, это позволяет ему запоминать элементы последовательности как на короткие, так и на длинные промежутки времени. Ключом к данной возможности является то, что LSTM-модуль не использует функцию активации внутри своих рекуррентных ячеек. Таким образом, хранимое значение не размывается во времени, и градиент или штраф функции потерь не исчезает при использовании метода обратного распространения ошибки [23] во времени при тренировке сети.

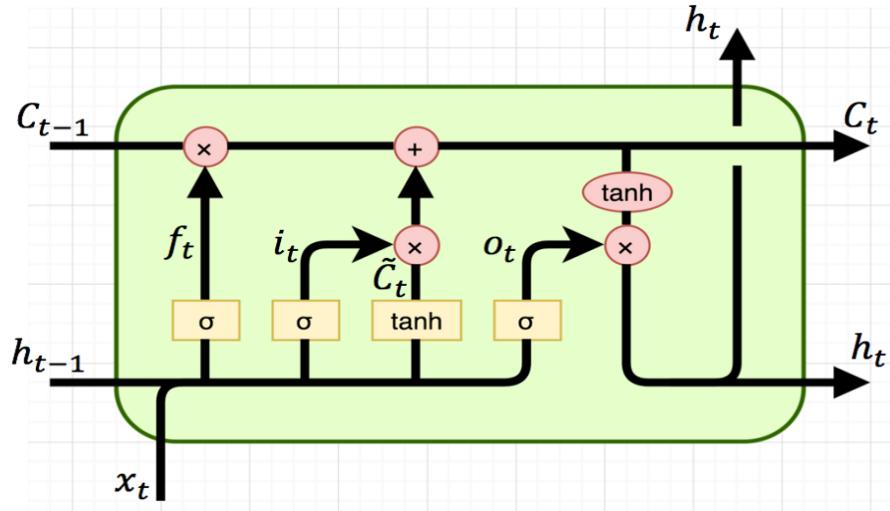


Рис. 10: Схематичное изображение развернутой временной ячейки сети с Долгой краткосрочной памятью [24].

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (9)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (10)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (11)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \quad (12)$$

$$h_t = o_t \circ \sigma_h(c_t) \quad (13)$$

где x_t - входной вектор (вектор элемента последовательности).

h_t - выходной скрытый вектор.

c_t - скрытый вектор состояний, который хранится внутри сети LSTM.

W, U, b - матрицы весов и добавочный вектор, единые для всех временных ячеек.

f_t - вектор вентиля забывания, вес запоминания старой информации.

i_t - вектор входного вентиля, вес получения новой информации.

o_t - вектор выходного вентиля, кандидат на выход.

σ_g - сигмоида, описанная в формуле (8).

σ_c, σ_h - функции на основе гиперболического тангенса (по сути просто являются функциями \tanh).

Таким образом данная сеть обучалась классифицировать последовательности и в качестве выхода выдавала число - вероятность класса 1 или 0, то есть вероятность того, купит ли последний продукт в последовательности покупатель или нет. В качестве функции потерь использовалась та же функция, что и для базовой модели. Этот подход позволил делать рекомендации так же, как и в базовой модели, то есть рекомендовать продукты с наибольшей вероятностью принадлежности к классу 1.

Другой подход из области обработки естественного языка был на основе так называемой парадигмы Sequence to sequence (Seq2Seq). Алгоритм для решения этой задачи был разработан в компании Google [25]. Данный алгоритм предназначен для решения таких задач, как машинный перевод, когда нужно одну последовательность перевести в другую. В нашей работе использовался алгоритм под названием Кодировщик-Декодировщик (Encoder-Decoder) с Механизмом Внимания (Attention) [26], который представляет собой два соединенных последовательно блока LSTM, между которыми вставлен слой Внимания.

В процессе обучения на первый блок (Кодировщик) подается исходная последовательность $x_t, x_{t+1}, x_{t+2}, x_{t+3}, \dots$, которая полностью проходит все временные ячейки и на каждом временном шаге $t, t+1, t+2, t+3\dots$ генерирует выходные вектора $h_t^{enc}, h_{t+1}^{enc}, h_{t+2}^{enc}, h_{t+3}^{enc}\dots$. Затем элемент y_t последовательности, которую

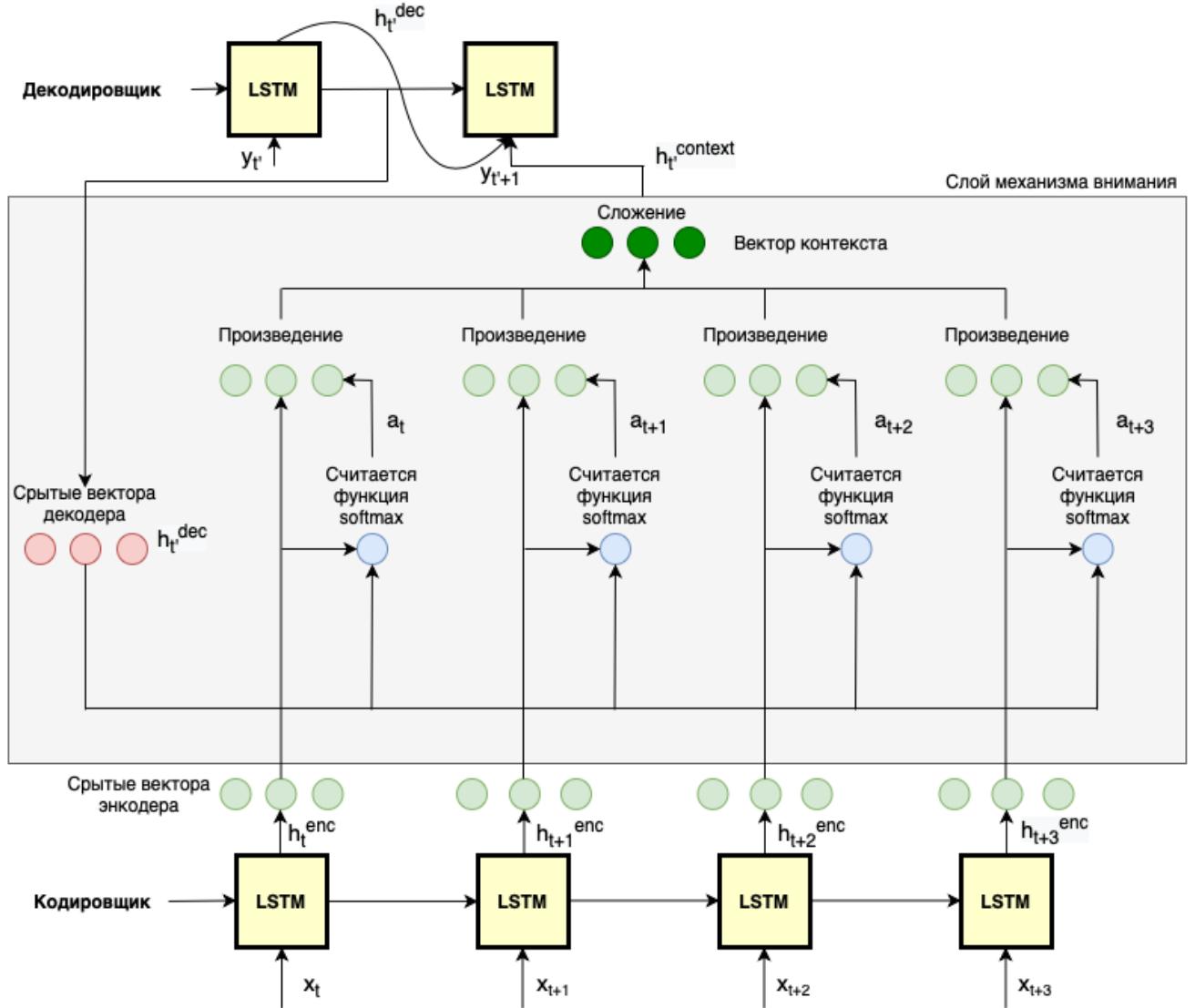


Рис. 11: Схематичное изображение алгоритма Кодировщик-Декодировщик с использованием Механизма Внимания.

мы хотим предсказать, подается на второй блок LSTM (Декодировщик). На временном шаге декодера t' генерируется выходной вектор $h_{t'}^{dec}$, для такого вектора считается скалярное произведение его со всеми выходными векторами энкодера $h_{t+1}^{enc}, h_{t+2}^{enc}, h_{t+3}^{enc} \dots$, затем по всем скалярным произведениям берется функция $softmax$:

$$softmax(x)_i = \frac{e^{-x_i}}{\sum_{k=1}^K e^{-x_k}} \quad (14)$$

где x - это вектор размерности K . Полученные числа $a_t, a_{t+1}, a_{t+2}, a_{t+3} \dots$ умножаются на выходные вектора энкодера $h_{t+1}^{enc}, h_{t+2}^{enc}, h_{t+3}^{enc} \dots$ и далее все вектора складываются, образуя вектор контекста

$h_{t'}^{context}$. Этот вектор контекста соединяется с вектором $h_{t'}^{dec}$ образуя новый вектор, который затем используется для того, чтобы предсказать следующий элемент последовательности $y_{t'+1}$. Данная процедура происходит для каждого выходного вектора декодера $h_{t'}^{dec}, h_{t'+1}^{dec} \dots$

Для обучения данного алгоритма составлялись пары последовательностей продуктов в двух соседних заказах для каждого пользователя. Таким образом модель пыталась переводить последовательность продуктов предыдущего заказа в последовательность продуктов следующего заказа. В качестве рекомендаций брались продукты из новой сгенерированной последовательности.

Еще один подход был на основе парадигмы предсказания следующего элемента последовательности. Это довольно распространенный способ обучения нейросетей типа LSTM, как раз этот тип рекуррентных сетей мы и использовали в своей работе. В качестве обучающей выборки брались все последовательности продуктов, купленных в один заказ для всех пользователей. Данная сеть учила предсказывать следующий купленный продукт в последовательности, таким образом решалась задача обучения без учителя. Такой подход можно интерпретировать как классификацию на N классов, где N - это количество уникальных слов (в нашем случае продуктов) в наборе данных. Такая интерпретация оправдывает себя, поскольку на каждом временном шаге t , модель в процессе обучения пытается максимизировать вероятность того, что на шаге $t + 1$ должно стоять одно из N слов (у нас продуктов), например k , то есть по сути это равносильно приписыванию последовательности, состоящей из элементов от 0 до t класса k , где $k = 0, \dots, N$. Поэтому в такой задаче используется функция ошибки мультиклассовая или категориальная кросс-энтропия (Categorical Cross-Entropy loss) [27], которая считается по формуле:

$$CE(x) = -\log(\text{softmax}(x)) \quad (15)$$

где softmax - это функция, описанная в формуле (14), а x - это вектор чисел, который выдает модель. В нашем случае вектор x обладал размерностью N .

Для получения рекомендаций на вход модели подавалась последовательность продуктов из предыдущего заказа и рекомендовались продукты с наибольшими вероятностями, которые модель генерировала на последней временной ячейке.

На самом деле под последовательностью продуктов в заказе подразумевается последовательность целочисленных уникальных индексов этих продуктов в словаре, который составляется изначально по всему набору данных. И поскольку нейросеть изначально не может работать просто с индексами, существуют некоторые способы по переводу индексов в вектора чисел. В нашей работе использовался так называемый слой векторных представлений слов (эмбеддингов), который изначально инициализировался произвольной матрицей размера количество уникальных слов в словаре на размерность вектора, которая является гиперпараметром и задается изначально. Таким образом во всех предыдущих методах перед тем, как подать последовательность на LSTM, она подавалась на слой векторных представлений, с которого на LSTM подавались уже вектора элементов последовательности. Этот

слой также обучался за счет метода обратного распространения ошибки. И затем вектора продуктов, обученных на этом слое добавлялись как дополнительные признаки на которых обучалась исходная базовая модель. Это делалось с целью улучшить ее качество.

Аналогично подходу, описанному в предыдущем абзаце, для получения векторных представлений продуктов мы также обучали модель Word2Vec [28], поскольку это достаточно несложный процесс, не требующий больших вычислительных ресурсов. Обучение происходило на той выборке, которая составлялась для подхода из предыдущего абзаца. Не вдаваясь в подробности процесса обучения данного алгоритма, стоит только обратить внимание на то, что он был создан именно с целью получать векторные представления за счет учитывания соседних слов в предложении, тем самым он переводит слова, стоящие близко друг к другу в предложении, в «близкие» вектора в новом векторном пространстве. Под словом «близкие» подразумевается близко расположенные или похожие с точки зрения некоторой меры, в частности с точки зрения косинусного расстояния. Размерность векторов является гиперпараметром и задается заранее.

Поскольку у нас в данных имелось текстовое название продуктов в файле products.csv, то имело смысл извлекать векторные представления этих названий из различных популярных предобученных моделей из области обработки естественного языка. Мы использовали модель из библиотеки под названием FastText, разработанной компанией Facebook's AI Research lab для извлечения векторных представлений из текста и для задач классификации. Внутри этой библиотеки Facebook предоставил множество предобученных моделей для извлечения векторных представлений на разных языках с помощью нейронных сетей. Алгоритм, использованный в этой модели был написан на основе статей исследователей из лаборатории Facebook AI Research [29]. Также использовалась еще одна модель, разработанная лабораторией Facebook AI Research. Эта модель называется LASER (Language-Agnostic SEntence Representations) [30], и была создана для задач машинного перевода и представляет собой несколько усложненную архитектуру Кодировщика-Декодировщика, о котором говорилось выше. В частности, одно из различий состоит в том, что в кодировщике в этой модели используется не просто рекуррентная сеть LSTM, а несколько идущих друг за другом двунаправленных сетей LSTM (Bidirectional LSTM) [31]. Их отличие от обычных сетей LSTM заключается в том, что для одной последовательности они считают скрытые состояния в одну и в другую стороны этой последовательности, то есть по сути это то же самое, что подать на обычную LSTM последовательность, она на каждом временном шаге посчитает выходной вектор, затем мы на эту же сеть LSTM подадим ту же последовательность, но в обратном порядке, также получим некоторые выходные вектора, затем соединим вектора, соответствующие одинаковым словам (элементам последовательности) в один, но удвоенного размера, таким образом получим тот же самый выход, что и с двунаправленной LSTM. Facebook расположил в открытом доступе обе предобученные модели LASER и FastText, которые мы использовали для извлечения векторных представлений из названий продуктов и добавления их в базовую модель.

Еще одна модель, которая использовалась для извлечения векторных представлений названий про-

дуктов, называется ELMo (Embeddings from Language Models) [32]. Эта модель разработана специально для получения векторных представлений слов с учетом их окружающего контекста и семантики. Отличительной чертой этого алгоритма является то, что в качестве входных данных в ней используются не последовательности слов, а букв, и при этом обучаются именно векторные представления букв. Данная модель успешно показала себя в классических задачах обработки естественного языка, поэтому имело смысл опробовать ее в нашей задаче.

Последней моделью, которую мы использовали, был BERT [33]. Эта модель создавалась для двух задач: для предсказания того, является ли одна последовательность продолжением другой или нет (то есть задача классификации на два класса), а также для заполнения пропущенных слов в предложениях. Однако эта модель показала лучшие результаты и во многих других классических задачах обработки естественного языка. Главным элементом архитектуры данной модели является так называемый Трансформер (Transformer) [12]. Он представляет собой Кодировщик-Декодировщик, основанный не на сетях LSTM, а на Механизме Внимания. Хотя BERT разрабатывался и не для получения векторных представлений слов, однако имеется возможность извлекать их из этой модели, а в связи с его большой популярностью, стоило попробовать применить его в нашей работе.

На основании векторных представлений продуктов, полученных с помощью моделей, описанных в предыдущих абзацах, мы составляли вектор контекста покупателя и также добавляли его в качестве новых признаков в базовую модель. Под вектором контекста подразумевается усредненный вектор последовательности продуктов, купленных в предыдущем заказе, то есть для каждого пользователя и для каждого его заказа мы брали последовательность продуктов из предыдущего его заказа, брали вектора этих продуктов, извлеченные с помощью описанных выше моделей, и усредняли их таким образом, чтобы получился один вектор той же размерности, этот вектор контекста мы добавляли в базовую модель и обучали ее. Помимо того, чтобы брать средний вектор только за предыдущий заказ, мы также брали усредненный вектор за все предыдущие заказы, то есть все последовательности продуктов, купленных в предыдущие заказы, вытягивались в одну последовательность, после чего вектора продуктов в этой последовательности усреднялись. На основании данного подхода независимо были проведены попытки рекомендовать покупателю продукты, вектора которых расположены близко к вектору контекста с точки зрения косинусного расстояния. Рекомендации осуществлялись так же как и в описании базовой модели, только весь рекомендуемый список продуктов фильтровался таким образом, что оставлялись только близкие к вектору контекста продукты.

Для всех подходов, описанных выше, использовались две целевые метрики под названиями Точность при k (precision at k) и Полнота при k (recall at k) [34]:

$$precision@k = \frac{N_k}{N_{rec}} \quad (16)$$

$$recall@k = \frac{N_k}{N_{rel}} \quad (17)$$

где N_k - это количество релевантных товаров, то есть количество товаров, которые покупатель действительно купил из k рекомендованных, N_{rec} - это общее число рекомендованных товаров, то есть по сути это число равно k , N_{rel} - это общее число релевантных товаров, то есть количество купленных покупателем товаров.

Таким образом все описанные выше модели и подходы измерялись по одинаковым метрикам, что позволило их сравнивать между собой.

2.3 Предобработка данных

В первую очередь были разделены заказы всех пользователей на заказы для обучения, заказы для валидирования и для тестирования моделей. Для этого по каждому пользователю брались все его заказы в той последовательности, в которой они были сделаны, и для тестирования выделялись два последних заказа, для валидации два предпоследних, то есть третий и четвертый с конца, все остальные заказы выделялись для обучения. Таким образом собирались три списка заказов (уникальных идентификаторов заказов): для обучения, валидации и теста. Затем проводилось уменьшение числа покупателей, это делалось исходя из ограничения вычислительных ресурсов, таким образом было отобрано только десять тысяч покупателей, причем таких, у которых не было пересечения по заказам из обучающей, валидационной и тестовой выборок, можно сказать, что это покупатели, которые сделали строго больше четырех заказов. Следующим шагом для выделенных пользователей был собран отдельный файл `order_features.csv`, в котором для каждого уникального идентификатора заказа была собрана последовательность продуктов, которые покупатель, которому соответствует данный заказ, купил в этот заказ, а также последовательность продуктов из предыдущего заказа, опять же, под последовательностью продуктов подразумевается последовательность уникальных идентификаторов продуктов. В дальнейшем этот файл `order_features.csv` был дополнен последовательностью продуктов, купленных за все предыдущие заказы для каждого пользователя и каждого его заказа. Это ключевой файл для обучения всех моделей, работающих с последовательностями.

Следующим шагом был сбор словаря, содержащего уникальные идентификаторы продуктов и переводащего идентификатор продукта в новый идентификатор-число, которое в дальнейшем будет использоваться для обучения моделей, работающих с последовательностями. Сбор производился только по заказам для обучения, а затем заказы для валидации и теста фильтровались согласно собранному словарю. Процесс фильтрации проходил следующим образом: для каждого обучающего заказа из файла `order_features.csv` в словарь добавлялись продукты из последовательностей в этих заказах, затем из того же файла брались заказы из валидационной и тестовой выборок, и для каждого такого заказа проверялось, все ли продукты из него содержатся в словаре, если нет, тогда пользователь, которому соответствовал данный заказ, помечался как «плохой» и добавлялся в список на удаление, далее изначальный список из десяти тысяч пользователей фильтровался по "плохим" пользователям, то есть из него удалялись плохие пользователи, и затем также заказы, соответствующие «плохим» пользователям

удалялись из файла `order_features.csv`, затем процедура сборки словаря аналогично повторялась по отфильтрованному файлу `order_features.csv`. Весь этот описанный цикл повторялся, пока в конечном итоге не осталось «плохих» пользователей, то есть все продукты из валидационной и тестовой выборок попали в словарь. В конечном итоге после всех процессов предобработки осталось 6605 пользователей, для них осталось 102474 обучающих заказов, 13210 валидационных заказов и 10705 тестовых заказов. Конечный размер словаря составил 24818 уникальных продуктов.

	<code>order_id</code>	<code>user_id</code>	<code>eval_set</code>	<code>order_number</code>	<code>order_dow</code>	<code>order_hour_of_day</code>	<code>days_since_prior_order</code>	<code>prod_seq</code>	<code>prev_seq</code>	<code>prev_orders_seq</code>
3	2254736	1	prior	4	4	7	29.0	[196, 12427, 10258, 25133, 26405]	[196, 12427, 10258, 25133, 30450]	[196, 14084, 12427, 26088, 26405, 196, 10258, ...]
4	431534	1	prior	5	4	15	28.0	[196, 12427, 10258, 25133, 10326, 17122, 41787...]	[196, 12427, 10258, 25133, 26405]	[196, 14084, 12427, 26088, 26405, 196, 10258, ...]
5	3367565	1	prior	6	2	7	19.0	[196, 12427, 10258, 25133]	[196, 12427, 10258, 25133, 10326, 17122, 41787...]	[196, 14084, 12427, 26088, 26405, 196, 10258, ...]
6	550135	1	prior	7	1	9	20.0	[196, 10258, 12427, 25133, 13032]	[196, 12427, 10258, 25133]	[196, 14084, 12427, 26088, 26405, 196, 10258, ...]
7	3108588	1	prior	8	1	14	14.0	[12427, 196, 10258, 25133, 46149, 49235]	[196, 10258, 12427, 25133, 13032]	[196, 14084, 12427, 26088, 26405, 196, 10258, ...]
8	2295261	1	prior	9	1	16	0.0	[49235, 46149, 25133, 196, 10258, 12427]	[12427, 196, 10258, 25133, 46149, 49235]	[196, 14084, 12427, 26088, 26405, 196, 10258, ...]
9	2550362	1	prior	10	4	8	30.0	[196, 46149, 39657, 38928, 25133, 10258, 35951...]	[49235, 46149, 25133, 196, 10258, 12427]	[196, 14084, 12427, 26088, 26405, 196, 10258, ...]

Рис. 12: Пример содержания файла `order_features.csv`. Поле `prod_seq` содержит последовательности продуктов, купленных в заказе с соответствующим идентификатором `order_id`, `prev_seq` содержит последовательности продуктов, купленных в предыдущем заказе относительно соответствующего `order_id`, `prev_orders_seq` содержит последовательности продуктов за все предыдущие заказы. Все остальные поля описаны в подразделе [Данные](#).

Следующим шагом была генерация признаков для обучения базовой модели. Первый признак это просто полное число заказов каждого пользователя `u_total_orders`. Следующий признак говорил о том, как часто покупатель в среднем повторно заказывает продукты, он назывался `u_reordered_ratio`. Для его создания использовалась метка `reordered` следующим образом, для каждого пользователя выбиралось сколько всего раз он совершил покупки, потом бралось число покупок (вернее число продуктов), в которые покупатель брал продукт, который он уже заказывал до этого, то есть с меткой `reordered` равной 1, и затем число покупок с меткой 1 делилось на общее число покупок. Далее генерировались признаки для продуктов. Первым из них был признак `p_total_purchases`, показывающий сколько всего раз данный продукт был куплен. Следующий признак - это доля повторных покупок продукта, то есть как часто его покупали повторно. Для этого также использовалась метка `reordered`, только теперь группировка происходила по продуктам, и количество повторных покупок продукта (`reordered` равняется 1) делилось на общее число его покупок. Далее формировались признаки, характеризующие взаимодействие конкретного покупателя с конкретным продуктом. Первым был признак, показывающий сколько раз данный покупатель купил данный продукт `uxp_total_bought`, здесь уже шла строгая привязка продукта к покупателю. Следующий признак `uxp_reordered_ratio` показывал как часто покупатель заказывал конкретный продукт после самого первого своего заказа этого продукта, то есть долю повторных заказов этого продукта после его первого заказа. Для этого считалось сколь-

ко раз всего заказал конкретный покупатель конкретный продукт. После этого определялось полное количество заказов, совершенных пользователем, и искался номер заказа у данного пользователя, в котором он первый раз купил данный продукт. Затем считалось сколько покупатель совершил заказов после первого заказа этого продукта учитывая сам этот первый заказ, и далее количество заказов, в которых покупатель брал конкретный продукт, делилось на количество заказов, сделанных после первого заказа этого продукта. Последний признак `times_lastN` говорил о том, сколько раз конкретный покупатель купил конкретный продукт за свои последние 5 заказов.

Все признаки генерировались только по заказам из обучающей выборки, соответственно вся статистика по взаимодействию покупателей с продуктами тоже собиралась по обучающей выборке. Однако так случилось, что после генерации признаков на обучающей выборке и присоединении данных признаков к оставшимся валидационным и тестовым заказам, некоторые значения признаков оказались незаполненными (то есть несуществующими со значением `NaN`). Это произошло потому, что в валидационной и тестовой выборке оказались продукты, с которыми некоторые покупатели не взаимодействовали (которые не покупали) в заказах из обучения. Таким образом возникала проблема «холодного старта», то есть необходимо было рекомендовать новые товары покупателям. Поскольку не хватало именно признаков взаимодействия покупателя с продуктом, то необходимо было заполнить значения признаков `uxp_total_bought`, `uxp_reordered_ratio` и `times_lastN`. Это осуществлялось следующим образом:

1. По каждому покупателю `user` и по каждому продукту `product`, с которым этот покупатель не взаимодействовал в обучающей выборке, отбираются покупатели отличные от `user` и взаимодействовавшие с данным продуктом `product`. Таким образом получается некоторая подвыборка X из данных, показывающая взаимодействие отличных от `user` покупателей, взаимодействовавших с продуктом `product`.
2. Затем выбирается некоторое число K соседей, то есть других покупателей отличных от `user`, это число равнялось либо 10, либо размеру получившейся подвыборки X , то есть количеству покупателей, отличных от `user` и взаимодействовавших с продуктом `product`. Число 10 бралось только тогда, когда размер подвыборки X был больше 10.
3. Далее на полученной подвыборке X на признаках покупателя `u_total_orders` и `u_reordered_ratio` обучается алгоритм K Ближайших Соседей [15]. Это алгоритм, который в процессе обучения запоминает обучающую выборку, а затем в качестве предсказания выдает похожие в признаковом пространстве объекты, для которого мы хотим сделать предсказание. "Похожие" имеется в виду близкие с точки зрения некоторой метрики или алгоритма. В нашем случае для поиска ближайших объектов использовался алгоритм Шаровое Дерево [35]. Этот алгоритм удобен для поиска ближайших точек в многомерном пространстве и поэтому хорошо подходит в качестве целевого алгоритма для K Ближайших Соседей.

4. Потом обученный алгоритм K Ближайших Соседей запускался на признаках u_total_orders и $u_reordered_ratio$ целевого покупателя $user$ и отбирались K ближайших покупателей с известными признаками uxp_total_bought , $uxp_reorder_ratio$ и $times_lastN$ для продукта $product$. Затем значения каждого признака усреднялись по K покупателям и полученные значения приписывались в качестве признаков взаимодействия покупателя $user$ и продукта $product$.

Таким образом были заполнены все пропуски взаимодействия покупателей с продуктами и создан файл `train_positive.csv`, содержащий все примеры покупки покупателем данный продукт в данный заказ, содержащий признаки покупателя, продукта и признаки взаимодействия покупателя с продуктом, а также метку класса `label`.

Последним шагом в предобработке данных была генерация отрицательных классов, то есть создание таких примеров заказов для каждого покупателя, в которые данный покупатель не купил какой-либо продукт. Так как изначально в наборе данных имелась информация только о том, что покупатель купил данный продукт в данный заказ, а для обучения базовой модели необходимо был два класса, то возникла необходимость в искусственном создании отрицательных примеров. Генерация отрицательных примеров проходила по следующему алгоритму:

1. Для каждого покупателя $user$ и для каждого продукта $product$ который он купил выделяются в отдельный список $orders$ все заказы, в которые покупатель $user$ купил продукт $product$.
2. Далее отбираются в отдельный список $target_orders$ все заказы, в которые покупатель $user$ не брал продукт $product$. Также определяется некий размер $size$ подвыборки, которая будет извлекаться из списка $target_orders$, этот размер это по сути количество отрицательных примеров для покупателя $user$ и продукта $product$. В лучшем случае этот размер $size$ равен количеству заказов $orders$, в которые покупатель $user$ купил продукт $product$, однако не всегда размер списка $target_orders$ позволяет извлечь из него подвыборку без повторений размера $size$, поэтому в случаях, когда это сделать невозможно $size$ выбирается как размер $target_orders$.
3. Далее из списка $target_orders$ извлекается случайным образом подвыборка размером $size$ и покупателю $user$ ставится в соответствие продукт $product$ и заказы в количестве $size$, в которые данный покупатель не брал данный продукт.

Таким образом собирался файл `train_negative.csv`, содержащий отрицательные примеры для всех пользователей и всех заказов, в том числе и валидационных и тестовых. В итоге получилось 1202304 положительных примеров и 1008946 отрицательных.

user_id	product_id	order_id	uxp_total_bought	uxp_reorder_ratio	times_lastN	u_total_orders	u_reordered_ratio	p_total_purchases	p_reorder_ratio	
509132	5616	46979	1671184	9	0.450000	3	20.0	0.699164	2318.0	0.666523
1198014	10855	13176	803384	2	0.833333	2	3.0	0.437500	13124.0	0.838464
465913	5248	20119	2392171	1	1.000000	1	4.0	0.051282	674.0	0.758160
748738	7570	12374	345037	1	1.000000	1	1.0	0.000000	3.0	0.000000
429864	4843	28204	1626548	3	0.090909	0	53.0	0.842227	2829.0	0.716154
1164019	6687	6933	415295	2	0.705833	1	2.0	0.088235	52.0	0.634615
708956	7908	42828	110469	2	0.222222	0	47.0	0.818182	1301.0	0.727902
539099	5435	41319	69621	1	0.500000	1	15.0	0.443089	272.0	0.525735
427787	4825	39928	67801	32	0.727273	5	44.0	0.790323	1816.0	0.737335
1102909	11240	29926	2031978	2	0.181818	0	25.0	0.729249	334.0	0.661677
595822	6603	21267	1634424	1	1.000000	1	1.0	0.000000	557.0	0.723519
746395	7554	45066	1535301	1	0.166667	0	13.0	0.549180	2582.0	0.729667
262274	2997	43295	745393	3	1.000000	3	4.0	0.337662	845.0	0.752663
423223	4424	6184	1253879	5	0.161290	0	64.0	0.780105	1181.0	0.772227
845085	9420	8712	618679	2	0.133333	0	44.0	0.666667	3.0	0.333333
order_number	order_dow	order_hour_of_day	days_since_prior_order	label						
7	4	22		6.0	0					
5	0	14		23.0	1					
1	1	18		0.0	0					
1	6	9		0.0	1					
15	5	19		6.0	0					
6	6	14		14.0	1					
36	4	6		10.0	0					
17	6	7		14.0	1					
20	1	10		2.0	0					
26	1	9		9.0	1					
5	4	12		3.0	0					
8	2	8		9.0	1					
1	0	9		0.0	0					
38	5	10		3.0	1					
8	1	16		3.0	0					

Рис. 13: Пример содержания объединенных файлов train_positive.csv и train_negative.csv, поле label означает купил ли конкретный покупатель user_id конкретный продукт product_id в конкретный заказ order_id (класс 1) или нет (класс 0).

3 Реализация предложенных моделей и подходов

Вся работа была проделана с помощью использования инструментария языка Python [36]. Весь анализ данных проводился в облачном сервисе компании Google под названием Google Colab [37], который предоставляет доступ ко многим библиотекам для обработки и анализа данных, а также удобный интерфейс для работы на основе Jupyter Notebook [38]. Для работы использовались следующие основные библиотеки: pandas [39], numpy [40], re [41], tqdm [42], pickle [43], gc [44], sklearn [45], tensorflow 1.0 [46], scipy [47], os [48], statsmodels [49], pylab [50].

Как было сказано выше, в качестве базовой модели использовался алгоритм градиентного бустинга над деревьями. Использовалась реализация данного алгоритма из модуля [51]. Сначала на сгенерированных признаках uxp_total_bought, uxp_reorder_ratio, times_lastN, u_total_orders, u_reordered_ratio, p_total_purchases, p_reorder_ratio, order_dow, order_hour_of_day, days_since_prior_order и метке label обучался алгоритм базовой модели и на валидационной выборке производился подбор оптималь-

ных гиперпараметров. Подбор производился по поиску по сетке значений параметров, оптимальная комбинация параметров определялась средним значением целевых метрик, то есть выбиралась та комбинация, при которой среднее значение двух метрик было наибольшее. Оптимизировались следующие параметры:

- *min_child_weight* - определяет минимальную сумму весов всех объектов в листе дерева, если на каком-то шаге сумма весов объектов в листе меньше *min_child_weight*, то дальнейшее разбиение этого листа не происходит. Значения для оптимизации: 1, 5.
- *gamma* - определяет минимальное уменьшение функции потерь при разбиении листа для дальнейшего его разбиения. Значения: 0, 1.
- *subsample* - доля объектов из обучающей выборки, на которых будут строиться деревья в ансамбле. Значения: 0.6, 0.75, 0.8.
- *colsample_bytree* - доля признаков, на которых будут обучаться деревья в ансамбле. Значения: 0.4, 0.6.
- *max_depth* - определяет максимальную глубину деревьев в ансамбле. Значения: 3, 5.

Таким образом были подобраны оптимальные значения параметров: $\text{min_child_weight} = 1$, $\text{gamma} = 0$, $\text{subsample} = 0.6$, $\text{colsample_bytree} = 0.4$, $\text{max_depth} = 3$. Также задавался параметр *num_boost_round*, который определял количество алгоритмов в ансамбле, он не оптимизировался по причине экономии вычислительных ресурсов и был задан изначально равным 10. Все эти значения параметров устанавливались в дальнейшем при использовании базовой модели во всех подходах.

В данной работе целевые метрики $\text{precision}@k$ и $\text{recall}@k$ считались при значениях $k = 1, \dots, 10$. Для всех подходов они считались примерно одинаково и реализация подсчета варьировалась незначительно в зависимости от используемой модели. В общем алгоритм подсчета целевых метрик выглядел следующим образом:

1. Для каждого покупателя и для каждого его заказа из валидационной или тестовой выборки выделялся отдельно список продуктов, которые покупатель купил в этот заказ.
2. Следующим шагом отбирались все продукты из обучающей выборки, по которым существовали признаки взаимодействия пользователя с продуктом, тут как раз и пригодились сгенерированные признаки при решении задачи «холодного старта», поскольку в валидационных и тестовых заказах некоторые покупатели брали новые товары, с которыми раньше не взаимодействовали, и не было возможности их рекомендовать из-за отсутствия признаков.
3. Затем к полученным признакам покупателя, продуктов и их взаимодействия присоединялись признаки заказа, для которого мы хотим сделать рекомендацию. На полученных признаках запускалась модель, и для каждого продукта она выдавала некую вероятность класса 0 и 1.

4. Далее продукты сортировались по вероятности класса 1 по убыванию и в зависимости от количества продуктов k , которые мы хотим порекомендовать, считались целевые метрики.

Таким образом для каждого пользователя и для каждого его заказа получались значения целевых метрик, 10 для $precision@k$ и 10 для $recall@k$ при $k = 1, \dots, 10$. В конце значения метрик усреднялись по количеству заказов из тестовой или валидационной выборках.

В подходе, в котором решалась задача классификации последовательностей продуктов на два класса использовалась следующая архитектура нейронной сети:

- Слой векторных представлений. Размерность векторов равна 13.
- Слой LSTM. Размер вектора скрытого состояния равен 10.
- Полносвязный слой с выходным размером 10 и функцией активации (нелинейностью), заданной как гиперболический тангенс.
- Полносвязный слой с выходным размером 2 (2 класса) и функцией активации $softmax$ (14).

Во всех слоях кроме слоя векторного представления параметр, который определяет способ инициализации весов в матрицах *kernel_initializer* был задан с помощью *he_uniform* инициализации [52]. В слое векторного представления использовался инициализатор из равномерного распределения. В качестве алгоритма оптимизации для данной модели использовался Adam [53] с параметром *learning_rate* = 0.00001. В качестве функции потерь использовалась бинарная кросс-энтропия (15), бинарная означает наличие только двух классов. Оптимизируемой метрикой была двоичная точность (binary accuracy) [54]. Она просто считает на скольких объектах алгоритм дал правильный ответ по отношению к количеству всех объектов.

Размерности слоя векторных представлений и скрытого вектора состояния были одинаковы для всех сетей и выбирались исходя из следующего эмпирического правила, описанного в блоге разработчика из компании Google [55]:

$$N_{emb} = \sqrt[4]{l_{dict}} \quad (18)$$

$$N_{hid} = \frac{2}{3}(N_{input} + N_{output}) \quad (19)$$

Где N_{emb} - размерность векторного слоя, l_{dict} - размер словаря (количество уникальных идентификаторов продуктов), N_{hid} - размерность скрытого вектора, N_{input} - количество нейронов на входном слое, N_{output} - количество нейронов на выходном слое.

Целевые метрики для этой модели считались таким образом, что в качестве признаков, на которых запускалась модель для предсказаний выступали последовательности продуктов, где последовательности продуктов, купленных в предыдущий заказ, в конец ставились продукты, которые нужно было

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, None)]	0
embedding (Embedding)	(None, None, 13)	437294
lstm_1 (LSTM)	(None, 10)	960
dense_tanh (Dense)	(None, 10)	110
dense (Dense)	(None, 2)	22
<hr/>		
Total params: 438,386		
Trainable params: 438,386		
Non-trainable params: 0		
<hr/>		

Рис. 14: Краткое резюме по архитектуре сети в задаче классификации последовательностей.

порекомендовать, то есть продукты, с которыми пользователь взаимодействовал в обучающей выборке, с учетом тех продуктов, для которых решалась проблема «холодного старта». Дальнейший алгоритм такой же, как и для базовой модели.

В подходе с обучением Кодировщика-Декодировщика использовалась следующая архитектура:

- Слой векторного представления с использованием маскировочной функции, которая позволяет обучать модель на небольших подвыборках (батчах) с последовательностями разных длин таким образом, что все последовательности приводятся к одной длине (в нашем случае к максимальной длине в подвыборке) путем добавления к последовательности фиктивного элемента (в нашем случае это было число 0), который в дальнейшем не учитывается при подсчете функции ошибки и следовательно ее градиента. Из-за этого фиктивного элемента число строк в матрице векторных представлений (количество элементов в словаре) увеличивалось на 1. Размер выходного вектора равнялся 13. Использовался равномерный инициализатор.
- LSTM слой Кодировщика с размером скрытого вектора равным 10.
- LSTM слой Декодировщика с размером скрытого вектора равным 10.
- Полносвязный слой с выходным размером 10 и функцией активации гиперболическим тангенсом.

- Полносвязный слой с выходным размером равным количеству элементов в словаре плюс 1, это нужно было потому, что на Декодировщике решается задача предсказания следующего элемента. Функция активации $softmax$ (14).

Во всех слоях, кроме слоя векторного представления, параметр, который определяет способ инициализации весов в матрицах $kernel_initializer$ был задан с помощью $he_uniform$ инициализации [52]. В слое векторного представления использовался инициализатор из равномерного распределения. В качестве алгоритма оптимизации для данной модели использовался Adam [53] с параметром $learning_rate = 0.00001$. В качестве функции потерь использовалась категориальная кросс-энтропия (15). Оптимизируемой метрикой была категориальная точность (categorical accuracy) [56]. Она работает так же, как и двоичная, то есть считает, на скольких объектах алгоритм дал правильный ответ, по отношению к количеству всех объектов, только для случая многих классов.

Для подсчета целевых метрик писался отдельный алгоритм генерации последовательности с помощью обученного Кодировщика-Декодировщика. Этот алгоритм принимал входную последовательность и максимальную длину выходной последовательности и генерировал также последовательность продуктов заранее заданной максимальной длины. При подсчете целевых метрик эта максимальная длина варьировалась от 1 до 10, собственно как и количество k продуктов, которые мы хотим рекомендовать. Соответственно метрики считались по следующему алгоритму: для каждого покупателя и для каждого его заказа из тестовой или валидационной выборок создавался отдельный список продуктов, которые покупатель купил в данный заказ, а также бралась последовательность продуктов, купленных в предыдущий заказ, затем для каждого значения числа рекомендуемых продуктов k это число и последовательность подавались на вход алгоритму генерации и на выходе получалась последовательность продуктов длины k , которая затем сравнивалась с заранее сохраненным списком купленных в данный заказ продуктов и таким образом участвовала в вычислении значений целевых метрик для каждого k .

В парадигме предсказания следующего элемента последовательности (продукта) использовалась модель со следующей архитектурой:

- Слой векторного представления с выходным размером 13 и равномерным инициализатором весов $kernel_initializer = uniform$.
- LSTM слой с выходным размером 13 и инициализатором $kernel_initializer = he_uniform$.
- Полносвязный слой с выходным размером равным количеству слов в словаре плюс 1 (опять же из-за добавления фиктивного нулевого элемента), с функцией активации $softmax$ (14) и инициализатором $kernel_initializer = he_uniform$.

В качестве алгоритма оптимизации для данной модели использовался Adam с параметром $learning_rate = 0.00001$. В качестве функции потерь использовалась категориальная кросс-энтропия (15). Оптимизируемой метрикой была категориальная точность (categorical accuracy).

Layer (type)	Output Shape	Param #	Connected to
input_decoder (InputLayer)	[(None, None)]	0	
input_encoder (InputLayer)	[(None, None)]	0	
embedding (Embedding)	(None, None, 13)	322673	input_encoder[0][0] input_decoder[0][0]
lstm_encoder (LSTM)	[(None, None, 10), (960		embedding[0][0]
lstm_decoder (LSTM)	[(None, None, 10), (960		embedding[1][0] lstm_encoder[0][1] lstm_encoder[0][2]
score_attention (Dot)	(None, None, None)	0	lstm_decoder[0][0] lstm_encoder[0][0]
attention_activation (Activation)	(None, None, None)	0	score_attention[0][0]
attention_weighted_dot (Dot)	(None, None, 10)	0	attention_activation[0][0] lstm_encoder[0][0]
context (Concatenate)	(None, None, 20)	0	attention_weighted_dot[0][0] lstm_decoder[0][0]
dense_tanh (TimeDistributed)	(None, None, 10)	210	context[0][0]
dense (TimeDistributed)	(None, None, 24821)	273031	dense_tanh[0][0]
<hr/>			
Total params: 597,834			
Trainable params: 597,834			
Non-trainable params: 0			

Рис. 15: Краткое резюме по архитектуре Кодировщика-Декодировщика в задаче перевода последовательности в другую последовательность.

Для подсчета целевых метрик использовался следующий алгоритм: для каждого покупателя и каждого его заказа из валидационной или тестовой выборок составлялся список продуктов, которые покупатель купил в данный заказ, и выделялась последовательность продуктов из предыдущего заказа, далее эта последовательность подавалась на модель, которая предсказывала вероятности для продуктов в словаре, далее эти продукты в словаре сортировались согласно вероятности по убыванию, и для рекомендаций брались первые k продуктов.

Последняя модель, которая была обучена в данной работе, это Word2Vec. Использовалась реализация этого алгоритма из библиотеки gensim [57]. Гиперпараметры подбирались на валидационной выборке с помощью поиска по сетке:

- wv_size - размерность выходных векторов, принимала значения: 50, 100, 150.
- $window$ - размер окна, внутри которого учитывается контекст соседних слов, принимал значения: 10 (средняя длина последовательностей продуктов в заказах), 8 (медианное значение длин последовательностей продуктов в заказах), 78 (максимальная длина последовательностей).
- min_count - минимальное число встречаемости элемента последовательности необходимое для

того, чтобы он был добавлен в словарь, принимал значения: 5, 1.

- *epochs* - количество эпох (проходов по всему набору данных), принимал значения: 25, 50.

В итоге финальные значения параметров были таковы: $wv_size = 100$, $window = 78$, $min_count = 1$, $epochs = 50$. Таким образом были получены векторные представления для всех продуктов.

Также для получения векторных представлений продуктов, на использовании которых основывались некоторые подходы, описанные в разделе Методология, брались следующие предобученные модели: BERT из библиотеки bert-embedding [58], ELMO из библиотеки tensorflow_hub [59], FastText из библиотеки fasttext [60], Laser из библиотеки laserembeddings [61].

Для использования векторных представлений для каждой из этих моделей были созданы словари соответствия уникального идентификатора продукта и его векторного представления, затем размерность этих векторных представлений была понижена до числа 13 (объяснения, почему была выбрана эта размерность как целевая, приведены выше (18)). Для понижения размерности использовался алгоритм Метод Главных Компонент [62], который находит в исходном векторном пространстве такие направления, проекция на которые исходных векторных представлений сохраняла бы дисперсию между этими представлениями, другими словами дисперсия проекций на эти направления должна быть максимальна. В конечном счете данный алгоритм сводится к сингулярному разложению матрицы векторных представлений продуктов и отбирает только несколько первых сингулярных чисел и соответствующие им размерности. В нашем случае конечной размерностью векторов была 13. В дальнейшем сжатые векторные представления продуктов из описанных выше моделей использовались для добавления в базовую модель как дополнительные признаки, также использовались для формирования контекста заказа и для рекомендации продуктов, близких к вектору этому контексту, эти подходы описаны в подразделе Методология.

4 Результаты

В результате проведенного исследования многие предложенные подходы себя абсолютно не оправдали. Так, у модели Кодировщика-Декодировщика, работающая в парадигме перевода одной последовательности продуктов в другую, за несколько различных начальных инициализаций и процессов обучения на 25 эпох обнаружилась тенденция к тому, что значение функции потерь на обучающей выборке изменяется примерно от 3.703 ± 0.005 до 3.043 ± 0.005 , значение функции потерь на валидационной выборке падало от 3.111 ± 0.005 до 2.561 ± 0.005 при этом метрика обучения категориальная точность практически не изменяла своего значения и для обучающей выборки колебалась от 0.089 до 0.095, а для валидационной вообще не изменяла своего значения и была на протяжении всего обучения равной 0.098 ± 0.001 . Также для модели, работающей в парадигме предсказания следующего элемента в последовательности, функция потерь на обучающей выборке для нескольких инициализаций процесса обучения на 30 эпох изменяла свои значения на обучающей выборке от 3.308 ± 0.015 до 3.000 ± 0.005 , а на валидационной от 2.777 ± 0.005 до 2.508 ± 0.005 , при этом значения метрики обучения категориальной точности для обучающей выборки изменились от 0.001 ± 0.001 до 0.008 ± 0.004 , а на валидационной от 0.000 ± 0.001 до 0.009 ± 0.003 . Из этих результатов был сделан вывод что данные модели совсем не обучаются, и дальнейшие попытки их применения для сравнения с базовой моделью были отброшены. Причиной таких результатов могло стать то, что в данной задаче совсем не важна последовательность продуктов в предыдущем заказе, а важен именно набор этих продуктов. Еще одной из возможных причин такого поведения моделей могло стать недостаточное количество эпох при обучении, но это нельзя было обойти в нашем случае из-за ограничения вычислительных ресурсов.

В подходе классификации последовательностей на два класса модель имела склонность быстро переобучаться, за 5 эпох функция потерь на валидации падала от 0.685 ± 0.001 до 0.684 ± 0.001 и затем начинала расти. Значения целевых метрик для этой модели приведены в таблице ниже:

k	1	2	3	4	5	6	7	8	9	10
Точность	0.249 ± 0.003	0.224 ± 0.002	0.212 ± 0.002	0.206 ± 0.002	0.203 ± 0.001	0.200 ± 0.003	0.199 ± 0.002	0.198 ± 0.004	0.198 ± 0.001	0.197 ± 0.001
Полнота	0.046 ± 0.001	0.081 ± 0.002	0.114 ± 0.003	0.145 ± 0.002	0.177 ± 0.001	0.205 ± 0.003	0.233 ± 0.002	0.258 ± 0.001	0.283 ± 0.003	0.308 ± 0.003

Таблица 1: Средние значения и доверительные интервалы при уровне значимости 0.05 целевых метрик модели в парадигме классификации последовательностей на два класса.

Как будет видно дальше, эти значения целевых метрик намного меньше значений у базовой модели. Это показывало то, что данный подход себя не оправдал. Одной из причин быстрого переобучения модели было то, что в обучающей выборке было много примеров с одинаковыми последовательностями,

то есть для одной последовательности продуктов из предыдущего заказа было много примеров положительных примеров, когда в конец данной последовательности ставился продукт из нового заказа, что в свою очередь не слишком сильно изменяло саму последовательность в целом.

Подходы, связанные с добавлением в базовую модель вектора контекста, полученные из векторов продуктов из предыдущего заказа также показали неудовлетворительные результаты, они либо не вносили никакого изменения, либо портили базовую модель и понижали значения целевых метрик в среднем на 5 – 6%.

Единственные сравнимые с базовой моделью результаты давали подходы с добавлением векторных представлений продуктов в качестве дополнительных признаков. В конечном итоге для получения векторных представлений продуктов использовались модели Word2Vec, FastText, ELMO, BERT, LASER. Для их сравнения было запущено 10 инициализаций и собрана статистика по целевым метрикам при различных значениях k , таким образом, для каждого значения k было собрано 10 значений метрики Точность и 10 значений метрики Полнота. Средние значения и доверительные интервалы при уровне значимости 0.05 для целевых метрик приведены в таблицах ниже:

Точность при k	1	2	3	4	5	6	7	8	9	10
Базовая модель	0.459± 0.002	0.42 ± 0.001	0.389± 0.002	0.364± 0.002	0.344± 0.002	0.327± 0.002	0.314± 0.002	0.302± 0.002	0.292± 0.002	0.283± 0.002
Word2Vec	0.458± 0.004	0.418± 0.003	0.387± 0.004	0.362± 0.004	0.342± 0.004	0.325± 0.004	0.312± 0.003	0.301± 0.003	0.291± 0.003	0.282± 0.003
FastText	0.459± 0.001	0.419± 0.002	0.388± 0.002	0.363± 0.002	0.343± 0.002	0.326± 0.002	0.312± 0.002	0.301± 0.002	0.291± 0.002	0.282± 0.002
ELMO	0.457± 0.003	0.418± 0.002	0.387± 0.002	0.362± 0.002	0.342± 0.003	0.325± 0.002	0.311± 0.002	0.3 ± 0.002	0.29 ± 0.002	0.281± 0.002
BERT	0.458± 0.002	0.418± 0.002	0.387± 0.002	0.362± 0.002	0.342± 0.002	0.325± 0.003	0.312± 0.003	0.3 ± 0.002	0.29 ± 0.002	0.281± 0.002
LASER	0.455± 0.003	0.417± 0.004	0.385± 0.004	0.361± 0.004	0.341± 0.004	0.324± 0.004	0.311± 0.004	0.3 ± 0.004	0.29 ± 0.003	0.281± 0.003

Таблица 2: Средние значения и доверительные интервалы при уровне значимости 0.05 моделей для значений целевой метрики Точность при k .

Далее необходимо было выбрать статистический критерий для проверки гипотез статистической значимости различий, а как известно многие критерии сравнения выборок из распределений исходят из предположения о нормальности этих распределений, поэтому сначала необходимо было проверить это предположение о нормальности. Для этого для каждого значения k считался тест Шапиро-Уилка

Полнота при k	1	2	3	4	5	6	7	8	9	10
Базовая модель	0.08 ± 0.0	0.136 ± 0.0	0.183 ± 0.001	0.223 ± 0.001	0.257 ± 0.001	0.287 ± 0.002	0.315 ± 0.002	0.341 ± 0.002	0.364 ± 0.002	0.386 ± 0.003
Word2Vec	0.08 ± 0.001	0.136 ± 0.001	0.182 ± 0.002	0.222 ± 0.002	0.256 ± 0.003	0.285 ± 0.003	0.314 ± 0.003	0.34 ± 0.003	0.363 ± 0.003	0.384 ± 0.004
FastText	0.08 ± 0.0	0.136 ± 0.001	0.182 ± 0.001	0.222 ± 0.001	0.256 ± 0.002	0.285 ± 0.002	0.313 ± 0.002	0.34 ± 0.003	0.363 ± 0.003	0.384 ± 0.003
ELMO	0.08 ± 0.001	0.136 ± 0.001	0.182 ± 0.001	0.221 ± 0.001	0.255 ± 0.002	0.285 ± 0.002	0.313 ± 0.002	0.339 ± 0.002	0.362 ± 0.002	0.384 ± 0.002
BERT	0.08 ± 0.0	0.136 ± 0.001	0.182 ± 0.001	0.221 ± 0.002	0.255 ± 0.002	0.285 ± 0.002	0.313 ± 0.002	0.339 ± 0.002	0.363 ± 0.003	0.384 ± 0.003
LASER	0.08 ± 0.001	0.136 ± 0.001	0.181 ± 0.002	0.221 ± 0.002	0.254 ± 0.003	0.284 ± 0.003	0.313 ± 0.003	0.339 ± 0.004	0.362 ± 0.004	0.384 ± 0.004

Таблица 3: Средние значения и доверительные интервалы при уровне значимости 0.05 моделей для значений целевой метрики Полнота при k .

[63]. Результаты Р-значений по этому критерию для значений целевых метрик базовой модели без добавления дополнительных признаков и с добавлением векторных представлений продуктов из описанных выше моделей приведены в таблицах ниже:

Точность при k	1	2	3	4	5	6	7	8	9	10
Базовая модель	0.620	0.138	0.271	0.929	0.981	0.483	0.284	0.413	0.604	0.521
Word2Vec	0.012	0.003	0.016	0.062	0.139	0.185	0.213	0.248	0.265	0.284
FastText	0.091	0.771	0.676	0.993	0.694	0.877	0.73	0.716	0.63	0.635
ELMO	0.698	0.028	0.087	0.104	0.172	0.38	0.732	0.818	0.643	0.791
BERT	0.426	0.363	0.255	0.644	0.817	0.827	0.829	0.851	0.847	0.85
LASER	0.0	0.015	0.025	0.039	0.092	0.217	0.37	0.555	0.735	0.806

Таблица 4: Р-значения для теста Шапиро-Уилка для метрики Точность при k .

Как видно из таблиц, большинство распределений значений целевых метрик можно считать за нормальные при уровне Р-значимости равном 0.05, однако не для всех значений можно принять нулевую

Полнота при k	1	2	3	4	5	6	7	8	9	10
Базовая модель	0.246	0.814	0.441	0.183	0.829	0.531	0.636	0.987	0.57	0.736
Word2Vec	0.012	0.01	0.38	0.465	0.154	0.326	0.402	0.902	0.943	0.997
FastText	0.645	0.75	0.352	0.383	0.372	0.75	0.707	0.294	0.38	0.11
ELMO	0.192	0.193	0.359	0.484	0.673	0.403	0.794	0.962	0.9	0.641
BERT	0.168	0.827	0.151	0.878	0.744	0.862	0.781	0.86	0.827	0.754
LASER	0.005	0.063	0.312	0.11	0.428	0.579	0.599	0.713	0.734	0.499

Таблица 5: Р-значения для теста Шапиро-Уилка для метрики Полнота при k .

гипотезу о нормальности распределений при таком уровне значимости, а значит и подсчет доверительных интервалов для них не является правильным решением, единственная причина, по которой можно оправдать данное действие это то, что в нашем случае нужно было проделать больше испытаний для этих значений целевых метрик, но ввиду ограниченных вычислительных ресурсов и времени не было возможности провести больше испытаний. Таким образом был выбран Парный t - критерий Стьюдента для зависимых выборок [64], этот критерий был выбран потому, что модели обучались на одних и тех же данных и проверялись на одинаковых наборах данных. Результаты попарного сравнения целевых метрик моделей, использующих векторные представления продуктов, с базовой моделью приведены в таблицах ниже:

Точность при k	1	2	3	4	5	6	7	8	9	10
Word2Vec	0.639	0.352	0.414	0.303	0.273	0.327	0.294	0.383	0.389	0.352
FastText	0.759	0.863	0.592	0.419	0.397	0.42	0.398	0.443	0.395	0.359
ELMO	0.178	0.29	0.23	0.202	0.194	0.221	0.165	0.228	0.191	0.19
BERT	0.281	0.37	0.178	0.116	0.129	0.17	0.162	0.204	0.185	0.232
LASER	0.079	0.142	0.091	0.124	0.103	0.108	0.138	0.137	0.136	0.153

Таблица 6: Р-значение Парного t-критерия Стьюдента для попарного сравнения значений целевой метрики Точность при k моделей, использующих векторное представление продуктов, с базовой моделью.

Как видно из результатов, представленных в таблицах, все различия статистически незначимы и следовательно добавление векторных представлений продуктов в модель в качестве дополнительных признаков никак не улучшило качество исходной базовой модели.

Полнота при k	1	2	3	4	5	6	7	8	9	10
Word2Vec	0.548	0.299	0.423	0.436	0.332	0.31	0.245	0.324	0.379	0.35
FastText	0.632	0.632	0.517	0.424	0.397	0.332	0.347	0.437	0.407	0.443
ELMO	0.262	0.487	0.179	0.155	0.168	0.164	0.146	0.242	0.243	0.213
BERT	0.293	0.26	0.201	0.162	0.157	0.219	0.208	0.366	0.366	0.386
LASER	0.159	0.194	0.113	0.136	0.089	0.079	0.134	0.181	0.2	0.253

Таблица 7: Р-значение Парного t-критерия Стьюдента для попарного сравнения значений целевой метрики Полнота при k моделей, использующих векторное представление продуктов, с базовой моделью.

5 Выводы

Интерпретировать результаты проведенных исследований можно по-разному. Возможный провал применения методов из области обработки естественного языка произошел вследствие того, что в данном наборе данных последовательности продуктов в заказах абсолютно не важны с точки зрения зависимых друг от друга во времени элементов цепочки, с которыми работают сети LSTM и им подобные, даже просто сам набор продуктов в заказах оказался незначимым, поскольку модель Word2Vec не учитывает последовательность элементов, а учитывает просто их наличие в одном предложении. Видимо, для данного набора данных не имело смысла применять методы для работы с последовательностями. Также еще одной причиной провала предложенных методов была высокая значимость сгенерированных признаков для обучения базовой модели. Анализируя результаты модели классификации последовательностей, можно сделать вывод о применимости данной модели в качестве начального приближения и в случае, когда сложно сгенерировать хорошие признаки для более простого алгоритма. В качестве общего вывода можно принять утверждение о том, что не надо «изобретать велосипед», когда есть хорошая базовая модель и когда есть возможность сгенерировать важные признаки для ее обучения.

В качестве дальнейших исследований можно попробовать применить подходы, описанные в данной работе, для других наборов данных, с другой структурой и отсутствием возможности генерации признаков.

Список литературы

- [1] Netflix: A Personalized Viewing Experience;. Library Catalog: digital.hbs.edu Section: Data. Available from: <https://digital.hbs.edu/platform-digit/submission/netflix-a-personalized-viewing-experience/>.
- [2] Zhang S, Yao L, Sun A, Tay Y. Deep Learning based Recommender System: A Survey and New Perspectives;52(1):1–38. Available from: <http://arxiv.org/abs/1707.07435>.
- [3] Recommendation System Series Part 1: An Executive Guide to Building Recommendation System;. Available from: <https://towardsdatascience.com/recommendation-system-series-part-1-an-executive-guide-to-building-recommendation-system-608f83e26>
- [4] Nwankpa C, Ijomah W, Gachagan A, Marshall S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning;Available from: <http://arxiv.org/abs/1811.03378>.
- [5] Многослойный персептрон · Loginom Wiki;. Available from: <https://wiki.loginom.ru/articles/multilayered-perceptron.html>.
- [6] Rosenblatt FF. PRINCIPLES OF NEURODYNAMICS. PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS;.
- [7] Kramer MA. Nonlinear Principal Component Analysis Using Autoassociative Neural Networks;37(2):11.
- [8] 5.1 Архитектура сверточной нейронной сети;. Library Catalog: studfile.net. Available from: </preview/6871496/>.
- [9] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation Applied to Handwritten Zip Code Recognition;1(4):541–551. Conference Name: Neural Computation.
- [10] Рекуррентные нейронные сети — Викиконспекты;. Available from: http://neerc.ifmo.ru/wiki/index.php?title=%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%80%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B5_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8.
- [11] Graves A, Liwicki M, Fernandez S, Bertolami R, Bunke H, Schmidhuber J. A Novel Connectionist System for Unconstrained Handwriting Recognition;31(5):855–868. Available from: <http://ieeexplore.ieee.org/document/4531750/>.
- [12] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention Is All You Need;Available from: <http://arxiv.org/abs/1706.03762>.
- [13] Barkan O, Koenigstein N. Item2Vec: Neural Item Embedding for Collaborative Filtering;Available from: <http://arxiv.org/abs/1603.04259>.

- [14] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed Representations of Words and Phrases and their Compositionality. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, editors. Advances in Neural Information Processing Systems 26. Curran Associates, Inc.;. p. 3111–3119. Available from: <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>.
- [15] Cover T, Hart P. Nearest neighbor pattern classification;13(1):21–27. Available from: <http://ieeexplore.ieee.org/document/1053964/>.
- [16] Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-based Recommendations with Recurrent Neural Networks;Available from: <http://arxiv.org/abs/1511.06939>.
- [17] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling;Available from: <http://arxiv.org/abs/1412.3555>.
- [18] Arora S, Warrier D. Decoding Fashion Contexts Using Word Embeddings;p. 6.
- [19] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed Representations of Words and Phrases and their Compositionality;Available from: <http://arxiv.org/abs/1310.4546>.
- [20] Instacart Market Basket Analysis;. Library Catalog: www.kaggle.com. Available from: <https://kaggle.com/c/instacart-market-basket-analysis>.
- [21] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System;p. 785–794. Available from: <http://arxiv.org/abs/1603.02754>.
- [22] Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J. LSTM: A Search Space Odyssey;28(10):2222–2232. Available from: <http://arxiv.org/abs/1503.04069>.
- [23] Rumelhart DE, Hintont GE, Williams RJ. Learning representations by back-propagating errors;p. 4.
- [24] Mani K. GRU’s and LSTM’s;. Library Catalog: towardsdatascience.com. Available from: <https://towardsdatascience.com/grus-and-lstm-s-741709a9b9b1>.
- [25] Nayak T, Ng HT. Effective Modeling of Encoder-Decoder Architecture for Joint Entity and Relation Extraction;Available from: <http://arxiv.org/abs/1911.09886>.
- [26] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate;Available from: <http://arxiv.org/abs/1409.0473>.
- [27] Deep Learning;. Available from: <https://www.deeplearningbook.org/>.
- [28] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed Representations of Words and Phrases and their Compositionality. In: Burges CJC, Bottou L, Welling M,

- Ghahramani Z, Weinberger KQ, editors. Advances in Neural Information Processing Systems 26. Curran Associates, Inc.;. p. 3111–3119. Available from: <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>.
- [29] Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching Word Vectors with Subword Information;Available from: <http://arxiv.org/abs/1607.04606>.
- [30] Artetxe M, Schwenk H. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond;Available from: <http://arxiv.org/abs/1812.10464>.
- [31] Cui Z, Ke R, Pu Z, Wang Y. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction;Available from: <http://arxiv.org/abs/1801.02143>.
- [32] Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, et al. Deep contextualized word representations;Available from: <http://arxiv.org/abs/1802.05365>.
- [33] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding;Available from: <http://arxiv.org/abs/1810.04805>.
- [34] Recall and Precision at k for Recommender Systems - Maher Malaeb - Medium;. Available from: https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54.
- [35] Dolatshah M, Hadian A, Minaei-Bidgoli B. Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces;Available from: <http://arxiv.org/abs/1511.00628>.
- [36] Welcome to Python.org;; Library Catalog: www.python.org. Available from: <https://www.python.org/>.
- [37] Google Colaboratory;; Library Catalog: colab.research.google.com. Available from: <https://colab.research.google.com/notebooks/intro.ipynb>.
- [38] Project Jupyter;; Library Catalog: jupyter.org. Available from: <https://www.jupyter.org>.
- [39] pandas - Python Data Analysis Library;; Available from: <https://pandas.pydata.org/>.
- [40] NumPy;; Available from: <https://numpy.org/>.
- [41] re — Regular expression operations — Python 3.8.3 documentation;; Available from: <https://docs.python.org/3/library/re.html>.
- [42] tqdm · PyPI;; Available from: <https://pypi.org/project/tqdm/>.
- [43] pickle — Python object serialization — Python 3.8.3 documentation;; Available from: <https://docs.python.org/3/library/pickle.html>.

- [44] gc — Garbage Collector interface — Python 3.8.3 documentation;. Available from: <https://docs.python.org/3/library/gc.html>.
- [45] scikit-learn: machine learning in Python — scikit-learn 0.23.1 documentation;. Available from: <https://scikit-learn.org/stable/>.
- [46] TensorFlow;. Available from: <https://www.tensorflow.org/>.
- [47] SciPy.org — SciPy.org;. Available from: <https://www.scipy.org/>.
- [48] os — Miscellaneous operating system interfaces — Python 3.8.3 documentation;. Available from: <https://docs.python.org/3/library/os.html>.
- [49] Introduction — statsmodels;. Available from: <https://www.statsmodels.org/stable/index.html>.
- [50] PyLab - SciPy wiki dump;. Available from: <https://scipy.github.io/old-wiki/pages/PyLab>.
- [51] XGBoost Documentation — xgboost 1.2.0-SNAPSHOT documentation;. Available from: <https://xgboost.readthedocs.io/en/latest/>.
- [52] He K, Zhang X, Ren S, Sun J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification;Available from: <http://arxiv.org/abs/1502.01852>.
- [53] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization;Available from: <http://arxiv.org/abs/1412.6980>.
- [54] tf.keras.metrics.binary_accuracy | TensorFlow Core v2.2.0;. Library Catalog: www.tensorflow.org. Available from: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/binary_accuracy.
- [55] Introducing TensorFlow Feature Columns;. Library Catalog: developers.googleblog.com. Available from: <https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html>.
- [56] tf.keras.metrics.categorical_accuracy | TensorFlow Core v2.2.0;. Library Catalog: www.tensorflow.org. Available from: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/categorical_accuracy.
- [57] gensim: topic modelling for humans;. Library Catalog: radimrehurek.com. Available from: <https://radimrehurek.com/gensim/models/word2vec.html>.
- [58] Lai G. bert-embedding: BERT token level embedding with MxNet;. Available from: https://github.com/imgarylai/bert_embedding.

- [59] TensorFlow Hub;. Library Catalog: www.tensorflow.org. Available from: <https://www.tensorflow.org/hub>.
- [60] fastText;,. Library Catalog: fasttext.cc. Available from: <https://fasttext.cc/index.html>.
- [61] yannvgn. laserembeddings: Production-ready LASER multilingual embeddings;,. Available from: <https://github.com/yannvgn/laserembeddings>.
- [62] Einasto M, Liivamagi LJ, Saar E, Einasto J, Tempel E, Tago E, et al. SDSS DR7 superclusters. Principal component analysis;535:A36. Available from: <http://arxiv.org/abs/1108.4372>.
- [63] THE SHAPIRO-WILK AND RELATED TESTS FOR NORMALITY;..
- [64] Student. The Probable Error of a Mean;6(1):1–25. Publisher: [Oxford University Press, Biometrika Trust]. Available from: <https://www.jstor.org/stable/2331554>.