|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ***Радиотехнический***

КАФЕДРА ***СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (РТ5)***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***Метрика скорости реакции поисковой системы на внезапные новостные события***

Студент \_\_РТ5-81Б\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ Ф.А. Сафонов**\_\_\_

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Ю.Е. Гапанюк**\_\_

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Ю.Н. Кротов\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2024 г.*

# **РЕФЕРАТ**

Расчётно-пояснительная записка квалификационной работы бакалавра содержит 102 страницы. С приложениями объем составляет 135 страниц. Работа включает в себя 26 таблиц и 30 иллюстраций. В процессе выполнения было использовано 46 источников.

Объектом разработки является модуль системы распознавания графических образов на изображении.

Данный программный модуль позволяет обрабатывать загружаемые в качестве входных данных изображения с целью распознавания на них графических образов и дальнейшей их классификации. Процесс распознавания происходит с помощью нескольких методов, для каждого из которых отображается отдельный результат.

Цель работы заключается в исследовании различных методов, моделей и библиотек машинного обучения для их дальнейшего анализа, сравнения, а также проведения экспериментов по их использованию при решении задачи распознавания и классификации графических объектов на изображении.

В процессе выполнения квалификационной работы бакалавра подробно изучена предметная область, рассмотрены наиболее применимые к выбранной задаче методы и модели, проведен их сравнительный анализ, определены метрики оценки качества, проведены эксперименты по реализации поставленной задачи распознавания, а также реализован программный продукт в виде модуля системы для распознавания и классификации образов на изображении.

Пояснительная записка содержит 3 приложения.

СОДЕРЖАНИЕ

[Реферат 2](#_Toc42208446)

[СОДЕРЖАНИЕ 3](#_Toc42208447)

[Введение 5](#_Toc42208448)

[1. Постановка задач разработки 7](#_Toc42208449)

[1.1 Постановка задач проектирования 7](#_Toc42208450)

[1.2 Описание предметной области 8](#_Toc42208451)

[1.3 Перечень функциональных возможностей 11](#_Toc42208452)

[1.4 Анализ данных 12](#_Toc42208453)

[1.4.1. Данные для обучения моделей распознавания 13](#_Toc10643380)

[1.4.2. Критерии качества набора данных 14](#_Toc10643380)

[1.4.3. Выбор набора данных для моделей распознавания 16](#_Toc10643380)

[2. Исследовательская часть 23](#_Toc42208467)

[2.1 Постановка задачи исследования 23](#_Toc42208451)

[2.2 Подбор решения задачи исследования 24](#_Toc42208451)

[2.3 Структура моделей распознавания 24](#_Toc42208451)

[2.4 ИНС для классификации объектов на изображениях 28](#_Toc42208451)

[2.5 Архитектуры свёрточных нейронных сетей классификации 32](#_Toc42208451)

[2.6 Модели распознавания объектов на изображениях 41](#_Toc42208451)

[2.7 Метрики оценки качества моделей распознавания 57](#_Toc42208451)

[2.8 Критерии сравнения моделей распознавания 62](#_Toc42208451)

[2.9 Сравнительный анализ моделей распонзавания 63](#_Toc42208451)

[2.10 Определение важности критериев 66](#_Toc42208451)

[2.11 Нормирование критериев 71](#_Toc42208451)

[2.12 Ранжирование вариантов моделей распознавания 7](#_Toc42208451)2

[2.13 Выбор моделей распознавания 78](#_Toc42208451)

[3. Конструкторско-технологическая часть 80](#_Toc42208472)

[3.1 Архитектура программного модуля](#_Toc42208451) 80

[3.2 Рабочее окружение 83](#_Toc42208451)

[3.3 Структура программного модуля 85](#_Toc42208451)

[3.4 Создание пользовательского интерфейса 87](#_Toc42208451)

[3.5 Источник данных](#_Toc42208451) 90

[3.6 Программный модуль 91](#_Toc42208451)

[3.6.1. Описание пользовательского сценария 91](#_Toc10643380)

[3.6.2. Демонстрация функциональных возможностей приложения 92](#_Toc10643380)

[Заключение 96](#_Toc42208477)

[Список использованных источников 98](#_Toc42208478)

[Приложение А Графические материалы 103](#_Toc42208479)

[Приложение Б Техническое задание 120](#_Toc42208480)

[Приложение В Программа и методика испытаний](#_Toc42208488) 127

ВВЕДЕНИЕ

В мире существует множество поисковых систем: Bing, Google, Яндекс. Рынок поисковых систем очень конкурентный: любая поисковая система стремится привлечь к себе как можно больше пользователей, чтобы иметь большую прибыль с показа рекламы. Именно поэтому важными аспектами любой поисковой системы являются релевантность и качество поисковых страниц на выдаче. Как правило, учитываются первые 5-10 страниц, ведь именно их увидит пользователь. Яндекс каждый год вкладывает огромные средства на улучшения алгоритмов ранжирования. Запросы пользователя бывают разной тематики, например: медицина, спорт, новости. Такие тематики называются поисковыми срезами. Например, одна команда может улучшать выдачу на спортивном срезе, другая на медицинском. В этой работе речь пойдет о мониторинге качества поисковой выдачи на новостном срезе. Это задача крайне важна, т.к. доля новостных запросов очень велика и Яндексу крайне важно иметь качественную выдачу, которая сможет в полной мере удовлетворить интересы пользователей. Например, если человек идет по улице и видит, что на улице пожар, он может захотеть узнать, что случилось. Для этого он напишет запрос в поисковую систему: «пожар в Москве что случилось». На такой запрос хорошая поисковая система должна правильно обработать документы-кандидаты, которые пойдут пользователю на выдачу. В данном случае поиску важно показать новые, актуальные, «свежие», как мы говорим, документы.

В жизни часто происходят незапланированные внезапные новостные события: аварии, случаи смерти, стихийные бедствия. Как правило, такие внезапные новостные событие сильно интересуют людей в первые секунды их происшествия. В таком случае задача поисковой системы как можно раньше актуализировать выдачу по тому или иному новостному событию.

В данной работе представлена разработка метрики качества поисковой выдачи на новостных запросах, связанных с каким-либо событием. Чтобы иметь возможность построить такую метрику было решено множество задач внутри компании. Для предсказания новостных запросов о внезапных новостных событиях на которые Яндексу важно иметь актуальную выдачу с первых секунд были использованы алгоритмы машинного обучения, СatBoost (Categorical Boosting), различные нейросетевые признаки, а также нейросеть LSTM (Long short-term memory) для анализа временных рядов динамики запроса. На различных этапах подсчета метрики были применены нейронные сети, а именно BERT (Bidirectional Encoder Representations from Transformers), реализовано множество Map Reduce вычислений на языке запросов YQL (разработка Яндекса, аналог SQL). Не обошлось и без регулярной ручной, асессорской разметки. Стоит отметить, что затраты на разметку были снижены во много раз благодаря машинному обучению.

1. Постановка задач разработки
   1. Постановка задач проектирования

Задача поисковой системы — предоставлять пользователю возможность быстрого доступа к необходимой ему информации при помощи поиска в обширной коллекции доступных данных. Одна из самых распространенных реализаций поисковых систем — веб сервисы поиска текстовой или графической информации во Всемирной паутине. В данной работе речь пойдет об одной из самых популярных поисковых систем — Яндекс Поиск.

Яндекс Поиск не стоит на месте: тысячи разработчиков, аналитиков и менеджеров ежедневно работают над тем, чтобы алгоритмы Яндекс Поиска стали лучше. Чтобы увидеть улучшения необходимо иметь мониторинги и графики, которые бы отражали эффективность работы разных компонент Поиска. По таким мониторингам можно сделать вывод о проделанной работе и доказать, что проделанная работа, действительно, улучшила Поиск по тому или иному аспекту.

За любым графиком должен стоять какой-то смысл измеряемой величины, например график качественных и релевантных страниц на выдачи или график доли пользователей, успешно решивших свою задачу благодаря нашей выдачи.

При работе над построением таких мониторингов необходимо сформулировать критерия к измеряемой величине и разработать алгоритм ее измерения. Например, если мы хотим иметь график, показывающий среднюю долю качественных страниц на всех запросах пользователя за день, необходимо сформулировать критерии качества страницы и придумать, как этот критерий измерять, имея доступный ограниченный набор ресурсов.

Благодаря мониторингам разработчики могут систематически отслеживать тот или иной улучшаемый параметр, именно по графикам на мониторингах делается вывод о пользе новых алгоритмов и принимаются итоговые решения.

Яндекс Поиск — масштабный сервис, поэтому задача улучшения Поиска декомпозируется на множество других подзадач, так или иначе связанных между собой. Часто бывает, что поисковая система работает неидеально только на запросах определенной категории. Такие категории запросов называются срезами. Запросы из каждого среза имеют свою специфику: например, на медицинские запросы важно показывать проверенные врачами статьи, чтобы пользователь не прочитал ложную информацию и не причинил себе вред, а на новостные запросы важно показывать недавно созданные страницы, чтобы пользователь получал актуальную информацию. Из-за таких особенностей Яндекс Поиск применяет различные алгоритмы ранжирования в зависимости от среза запроса.

В данной работе речь пойдет о создании мониторинга для отслеживания качества выдачи Яндекс Поиска на ультрасвежих запросах.

Запрос является **ультрасвежим**, если с запросом связано какое-то новостное событие и соблюдаются следующие пункты:

1. Событие по запросу незапланировано. Незапланированным считаем событие, о котором не было информации до дня запроса.
2. Для объекта запроса произошедшее событие значимо. Значимым считаем событие, заметно выходящее за рамки того, что обычно происходит с объектом запроса.

Примеры незапланированных и значимых событий:

* ЧП
* Катаклизм
* Неожиданные заявления
* Происшествия
* Неожиданная смерть

В данной работе ставится задача разработать программный модуль для определения качества поисковой системы на таких запросах. Поскольку Поиску крайне важно быстро реагировать на внезапные события, качество особенно важно в первое время с момента происшествия события.

Таким образом, в результате анализа целевой направленности разработки, можно сформулировать следующие задачи выполнения данной выпускной квалификационной работы бакалавра:

* Изучение и анализ предметной области метрик алгоритмов ранжирования
* Определение критериев к запросам, на которых будет подсчитана желаемая метрика
* Создания алгоритма извлечения необходимых запросов, среди общего множества поисковых запросов
* Создание алгоритмов определения начала события, связанного с запросом
* Создания алгоритма определения качества выдачи Поисковой системы Яндекс в первые часы с момента происшествия события по данному запросу
  1. Описание предметной области

Рассмотрим основные понятия и определения, касающиеся данной предметной области:

Ранжирование — это сортировка сайтов поисковыми системами и размещение их в результатах выдачи

Метрика — это качественный или количественный показатель, который отражает ту или иную характеристику и уровень успешности продукта.

Поисковый запрос — последовательность символов, которую пользователей вводит в поисковую строку, чтобы найти интересующую его информацию. Формат поискового запроса зависит как от устройства поисковой системы, так и от типа информации для поиска. Чаще всего, поисковый запрос задаётся в виде набора слов или фразы.

Поисковая выдача — это список документов, найденных и проиндексированных поисковой системой. Обычно он упорядочен по убыванию релевантности документов поисковому запросу согласно применяемым в поисковой системе алгоритмам ранжирования, но во многих системах предусмотрены также и другие виды сортировки, например, по дате документов.

Релевантность — степень смыслового соответствия результатов выдачи запросу пользователя.

Другие термины, придуманные командой, чтобы определять нужные нам понятие будут разъяснены по ходу изложения.

1. Исследовательская часть
   1. Метрики ранжирования

Ранжирование — это классическая задача машинного обучения, которая заключается в ранжировании списка элементов на основе их релевантности для конкретной задачи (например, ранжирование страниц в Google на основе их соответствия заданному запросу). Задача имеет широкий спектр применений в различных сервисах:

* Рекомендации фильмов и видео (Кинопоиск, YouTube)
* Ранжирование страниц в поиковых системах (Yandex, Google, Bing)
* Рекомендации товаров (Ozon, Amazon)
* Поиск жилья (Booking, Yandex travel)

Чтобы измерить качество системы ранжирования, необходимо знать истинно верный порядок элементов и выбрать соответствующую задаче метрику.

Вот основные метрики качества ранжирования:

* MRR (Mean reciprocal rank)
* Precision@ K
* DCG & NDCG
* MAP
* Kendall’s tau
* Spearman’s rho

Рассмотрим основные из них (первые 3).

* 1. Mean reciprocal rank

Название этой метрики переводится как средний обратный ранг. Ранжирование работает тем лучше, чем ближе к началу выдачи релевантный для пользователя документ. Для каждого запроса найдём позицию первого релевантного документа, возьмём обратное от этого числа и усредним по всем запросам.

где:

— множество запросов

— первый релевантный эдемент

Чтобы проиллюстрировать это, давайте рассмотрим приведенный ниже пример, в котором модель пытается предсказать форму множественного числа английских слов, делая 3 предположения. В каждом случае также дается правильный ответ.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Запрос*** | ***Предсказания модели*** | ***Правильный ответ*** | ***Ранг*** | ***Обратный ранг*** |
| Child | Childs, **Children**, Child | Children | 2 | 1/2 |
| Medium | Medium, Meds, **Media** | Media | 3 | 1/3 |
| Virus | **Viruses**, Viri, Virii | Viruses | 1 | 1 |

MRR в таком случае будет равен:

Одним из ограничений MRR является то, что он учитывает только позицию первого релевантного элемента и игнорирует другие. Это может быть неподходящей метрикой для случаев, когда мы хотим оценить весь список элементов.

* 1. DCG & NDCG

Normalized Discounted Cumulative Gain (NDCG), пожалуй, самая популярная метрика качества моделей ранжирования. В отличие от предыдущих метрик, NDCG учитывает порядок и относительную важность документов и поощряет размещение наиболее релевантных документов как можно выше в списке предсказания.

Прежде чем дать определение NDCG, давайте сначала определим два показателя: Cumulative Gain (CG) и Discounted Cumulative Gain (DCG).

Cumulative Gain (CG) набора элементов представляет собой сумму их оценок релевантности и определяется следующим образом:

где:

— множество элементов, предсказанное моделью.

Предполагается, что релевантность каждого элемента посчитана.

Discounted Cumulative Gain (DCG) — это, по сути, взвешенная версия CG, в которой используется логарифмический понижающий коэффициент, значение которого зависит от позиции элемента в предсказании модели. Это полезно, так как на практике ы хотим придать более высокий приоритет первым нескольким элементам при анализе качества ранжирования. DCG определяется как:

где:

— множество элементов, предсказанное моделью.

— релевантность i-го элемента.

Normalized Discounted Cumulative Gain (NDCG) нормализованная версия метрики DCG. Поскольку предсказанное множество элементов может отличаться по размеру в зависимости от запросов или моделей возникает необходимость нормализовать получившееся значение посчитанной метрики DCG (путем деления на DCG идеальной системы). Можно сказать, что при подсчете идеального IDCG считается DCG элементов, отсортированных по релевантности. Формула, по которой считается метрика:

Где:

— множество элементов, предсказанное моделью.

— DCG «идеального» ранжирования модели.

NDCG является одной из самых популярных метрик, но несмотря на это, имеет и недостатки. Основной ее недостаток в том, что она не штрафует за неприемлемые элементы в получившемся ранжировании.

* 1. Рекурентные нейронные сети

Каждая следующая мысль человека, как правило, связана с предыдущей. Читая какой-либо материал, человек понимает каждое слово, основываясь на своем понимании предыдущих слов. При чтении каждой главы статьи человек держит в голове знания, полученные в предыдущих главах, чтобы лучше понять текущую. Мысли отличаются постоянством.

Обычные нейронные сети не могут эффективно извлекать информацию из прошлого, и это кажется серьезным недостатком. Например, представим, что необходимо классифицировать, какие события происходят в каждом моменте фильма. Неясно, как обычная нейронная сеть могла бы использовать полученную информацию о предыдущих событиях в фильме для классификации последующих.

Рекуррентные нейронные сети решают эту проблему. Это сети с циклами, которые умеют сохранять информацию о прошлом.

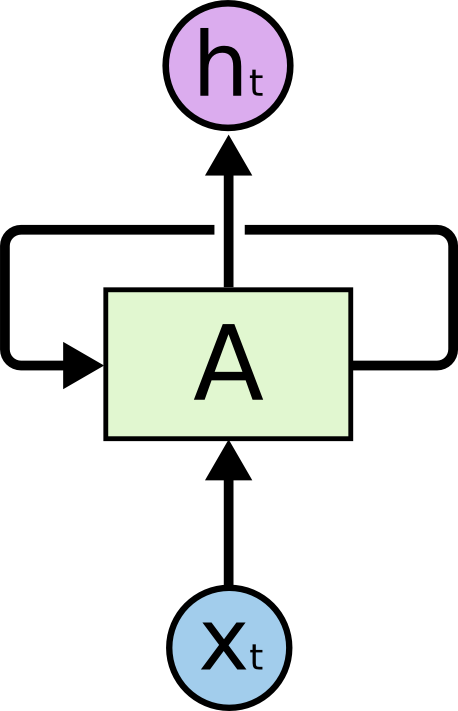


Рисунок 1 — фрагмент нейронной сети

На рисунке 1 представлен фрагмент нейронной сети. Блок A смотрит на входные данные и возвращает выходное состояние нейронной сети . Под блоком в данному случае подразумевается какой-то набор слоев нейронной сети. Цикл над блоком A позволяет нейронной сети получить информацию о предыдущих итерациях.

Рекуррентную нейронную сеть можно рассматривать как множество копий одной и той же сети, каждая из которых передает сообщение следующей. Таким образом, если искусственно развернуть цикл, получится следующая интерпретация:

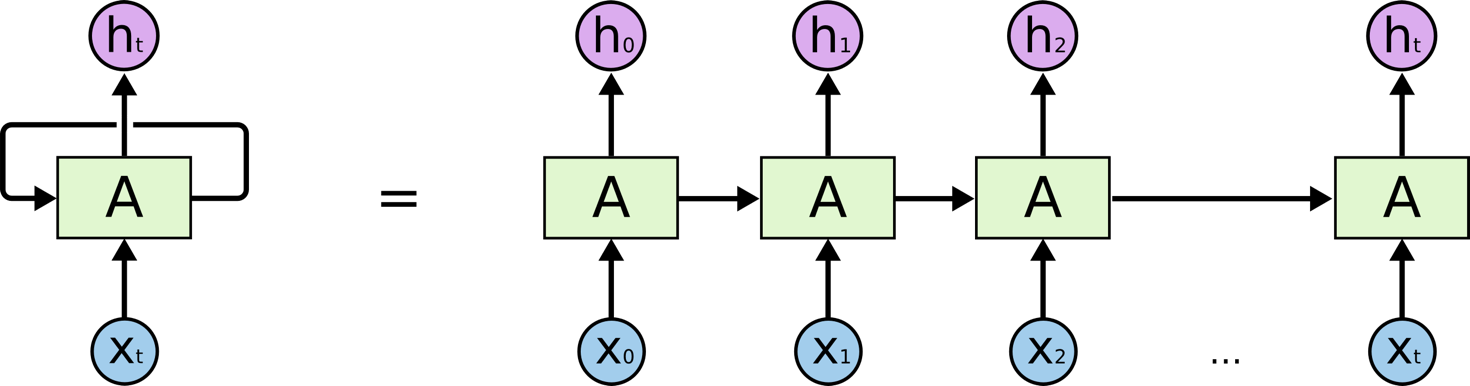


Рисунок 2 — рекуррентная нейронная сеть с развернутым циклом

Схема в виде цепи (Рисунок 2) показывает, что рекуррентные нейронные сети тесно связаны с обработкой последовательностей и списков. Это подходящая архитектура нейронной сети, используемая для обработки таких данных.

Одно из преимуществ рекуррентных нейронных сетей заключается в том, что они могут связать предыдущую информацию с текущей задачей, например, использование предыдущих видеокадров может помочь в понимании текущего кадра. Если бы рекуррентные нейронные сети могли это делать, они были бы чрезвычайно полезны. Но могут ли они? Зависит от контекста.

Иногда, нам необходимо посмотреть лишь на определенный фрагмент в прошлом. Например, рассмотрим языковую модель, которая пытается предсказать следующее слово на основе предыдущих. Если мы пытаемся угадать последнее слово в фразе “облака плывут по *небу*”, нам не нужен дополнительный контекст – совершенно очевидно, что следующим словом будет "небо". В таких случаях, когда разрыв между релевантной информацией и местом, где она необходима, невелик, рекуррентные нейронные сети могут эффективно использовать информацию о прошлом (Рисунок 3).

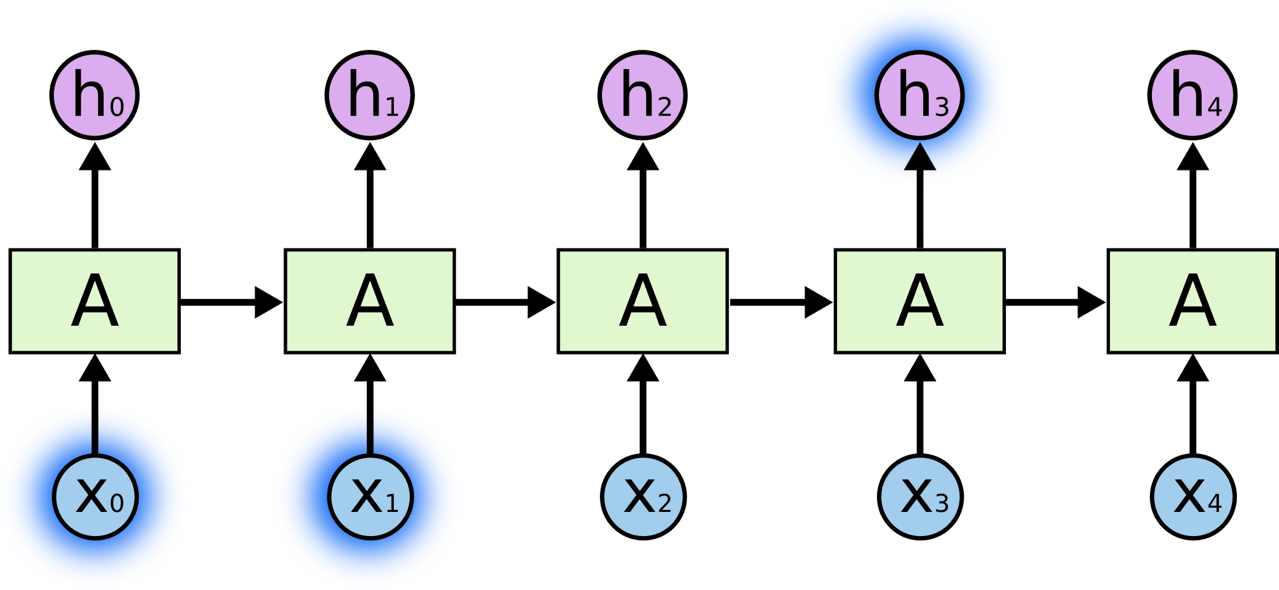


Рисунок 3 — рекуррентная нейронная сеть эффективно использует информацию о прошлом

Но бывают случаи, когда нам нужен более детализированный контекст. Попробуйте угадать последнее слово в тексте: “Я вырос во Франции… Я свободно говорю *по-французски*”. Последняя информация из контекста указывает на то, что следующее слово, вероятно, является названием языка, но если мы хотим уточнить, на каком именно языке, нам нужно не потерять информацию о Франции, которая встречалась в начале текста. Вполне возможно, что разрыв между релевантной информацией и местом, где она необходима, может оказаться слишком большим. К сожалению, по мере того, как этот разрыв растет, рекуррентные нейронные сети становятся неспособными связывать информацию воедино (Рисунок 4).

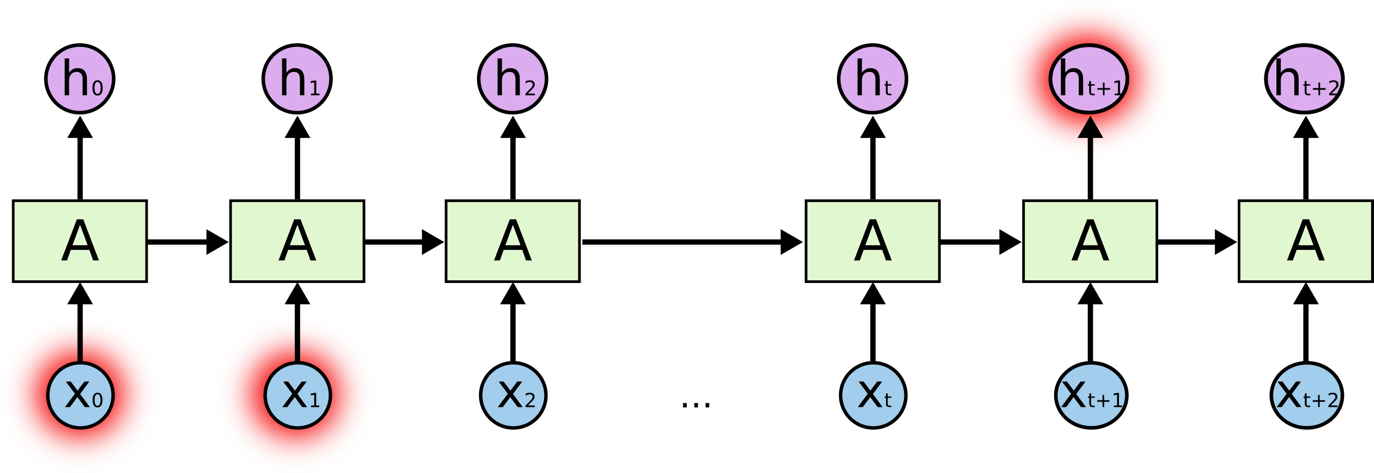


Рисунок 4 — рекуррентная нейронная сеть теряет информацию о прошлом из-за большого разрыва

Существуют другие архитектуры нейронных сетей, которые способны решить эту проблему.

* 1. LSTM

Рекуррентная нейронная сеть архитектуры Long Short Term Memory (LSTM) специально разработана для того, чтобы избежать проблемы отсутствия долговременной памяти. В отличие от обычных рекуррентных сетей, LSTM запоминает информацию, пришедшую спустя длительный период времени.

Все рекуррентные нейронные сети имеют форму цепочки повторяющихся ячеек нейронной сети. В стандартных RNNS эта повторяющаяся ячейка будет иметь очень простую структуру, например, один линейный слой с функцией активации tanh (Рисунок 5).

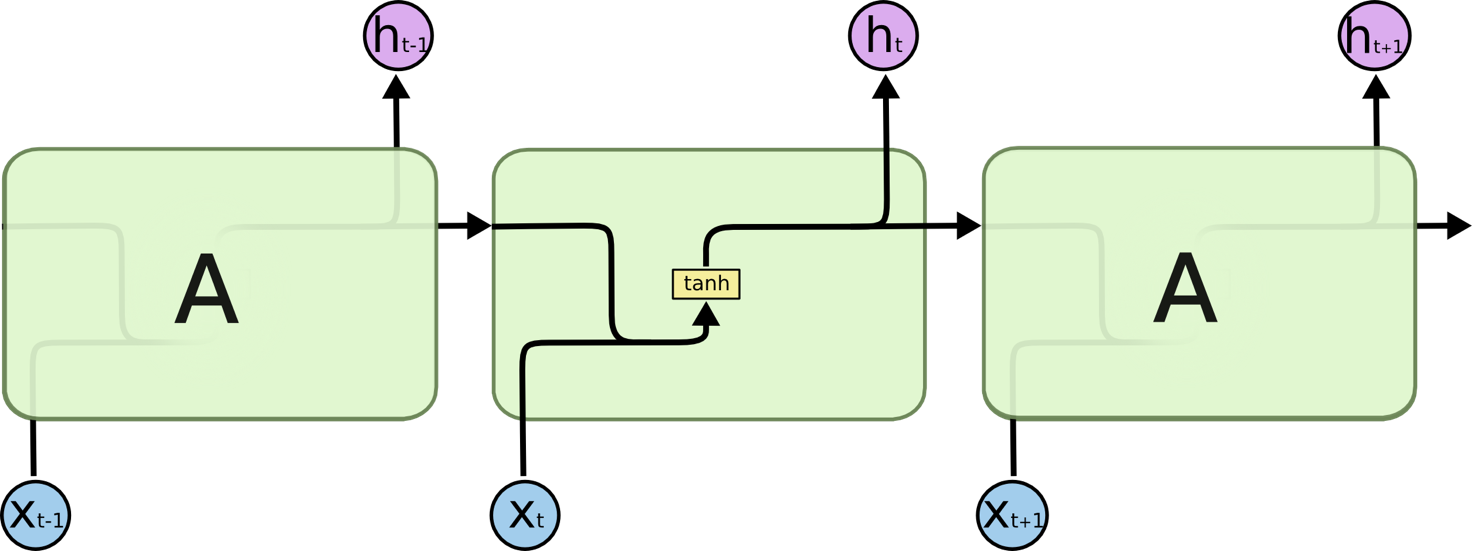


Рисунок 5 — повторяющиеся ячейки в обычной рекуррентной нейронной сети

LSTM также имеют цепочечную структуру, но повторяющаяся ячейка имеет другую структуру. Вместо одного слоя нейронной сети их четыре, которые взаимодействуют между собой особым образом (Рисунок 6).

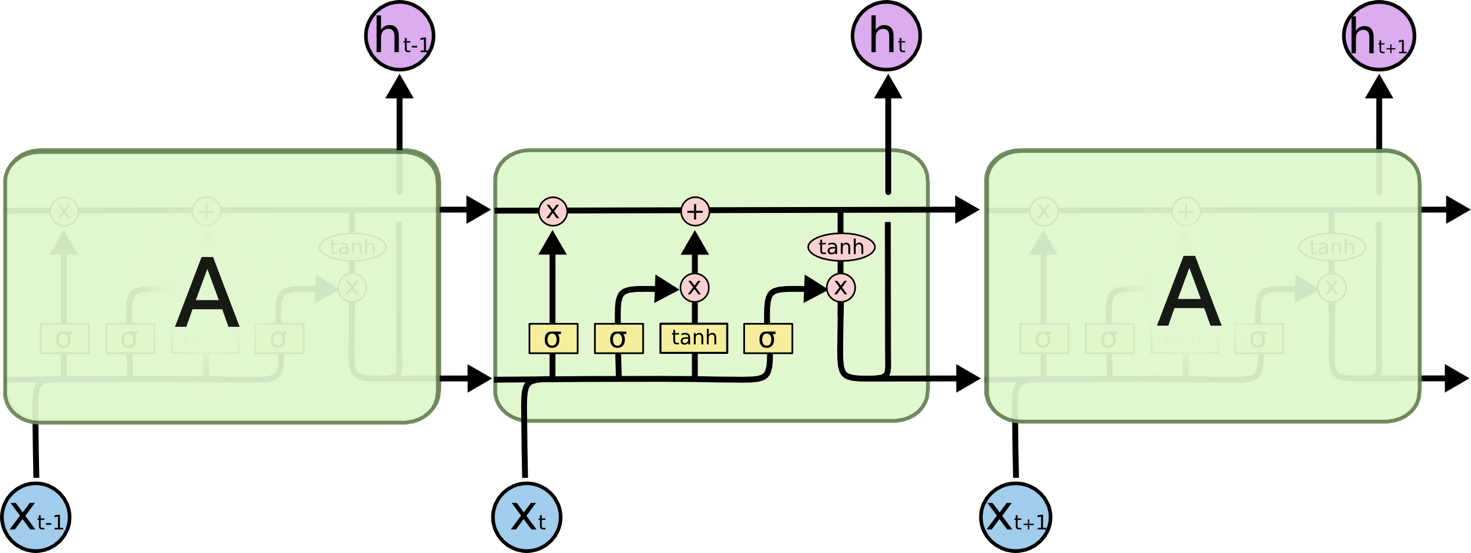


Рисунок 6 — Повторяющиеся ячейки в архитектуре LSTM

На приведенной выше схеме каждый элемент ячейки определяет определенное действие со входным вектором. Розовые кружки обозначают векторные операции, такие как сложение векторов, в то время как желтые прямоугольники обозначают обучаемые слои нейронной сети. Слияние линий означает объединение в единый вектор, в то время как разветвление строк означает, что их содержимое копируется и копии отправляются в разные места (Рисунок 7).



Рисунок 7 — Обозначение элементов ячейки LSTM

Ключевой особенностью LSTM является состояние ячейки (cell state), горизонтальная линия, проходящая через верхнюю часть диаграммы. Состояние ячейки похоже на ленту конвейера. Оно проходит прямо по всей ячейки, лишь с некоторыми незначительными линейными взаимодействиями (Рисунок 8).



Рисунок 8 — Состояние ячейки LSTM

LSTM может добавлять или удалять определенную информацию из cell state с помощью специальных механизмов, которые называются **gates** (ворота или вентили в русскоязычной литературе).



Рисунок 9 — Sigmoid слой гейта

На выходе после слоя с функцией активацией (ФА) Sigmoid получаются числа от нуля до единицы, описывающие, какое количество информации должно пройти к дальше по сети. Значения, близкие к нулю, означают “дальше информацию не пропускать ”, в то время как значения, близкие к единицам, означают “пропускать все!”.

Всего в LSTM 3 гейта, каждый из которых защищает и контролирует текущее состояние ячейки.

Первым шагом во время работы LSTM является принятие решения о том, какую информацию мы собираемся удалить из состояния ячейки (Рисунок 10). Это определяет выходная информация после слоя с ФА Sigmoid, который называется «forget gate layer». В этом гейте считывается информация и и вычисляются числа от 0 до 1 для каждого числа из сell state: 0 обозначает полностью забыть это число, 1 обозначает сохранения числа в памяти для дальнейшего использования.

Вернемся к нашему примеру с языковой моделью, которая предсказывает следующее слово на основе всех предыдущих. В такой задаче состояние ячейки может включать род существительного, чтобы можно было использовать правильные местоимения. Когда мы видим новый объект, мы хотим забыть пол старого объекта.



Рисунок 10 — Forget gate

Следующий шаг — решить, какую новую информацию мы будем сохранять в состоянии ячейки (Рисунок 11). Этот шаг состоит из двух частей. Сначала сигмоидальный слой, называемый «input gate layer», решает, какие значения мы будем обновлять. Затем слой с ФА tanh создает вектор новых возможных значений , которые могут быть добавлены в текущее состояние ячейки. На следующем шаге мы объединим эти два параметра, чтобы создать обновление состояния.

В примере с нашей языковой моделью мы хотели бы добавить пол нового субъекта в состояние ячейки, чтобы заменить старый, который мы забыли.

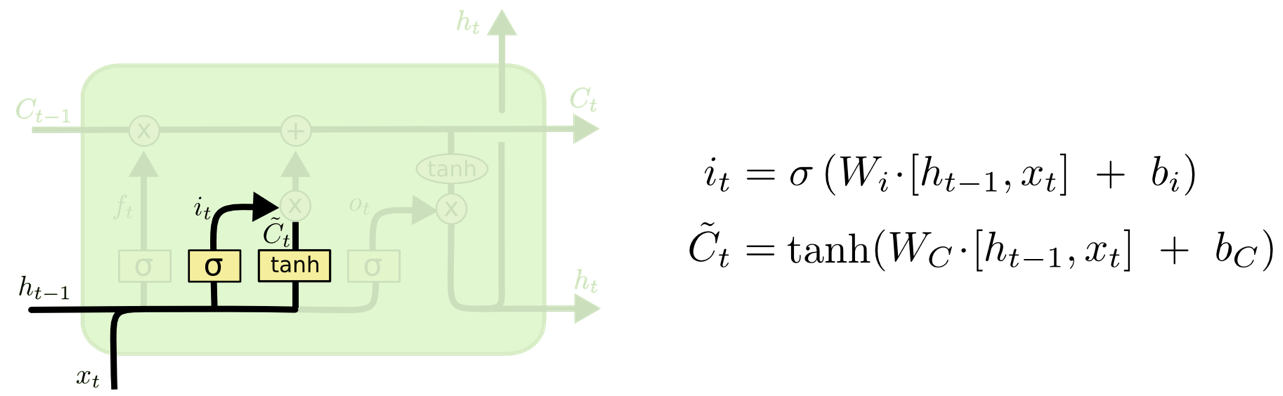


Рисунок 11 — Input gate

Теперь пришло время изменить старое состояние ячейки, , на новое состояние ячейки . На предыдущих шагах уже было посчитано, какую информацию удалить, а какую добавить, остается лишь сделать это.

Мы умножаем старое состояние на , удаляя ненужную информацию, которую решили удалить ранее. Затем мы добавляем *,* Это новые значения-кандидаты, умноженные на коэффициенты, определяющие силу обновления текущего числа скрытого состояния ячейки (Рисунок 12).

В случае с языковой моделью именно здесь мы фактически удалили информацию о поле старого субъекта и добавили новую информацию, посчитанную на предыдущих шагах.



Рисунок 12 — Обновление текущего состояния ячейки LSTM

Наконец, нам нужно решить, что мы будем отправлять в качестве выхода нейронной сети. Этот вывод будет основан на состоянии нашей ячейки, но это будет отфильтрованная версия. Сначала получаем выход сигмоидального слоя, который определяет, какие части состояния ячейки мы будем отправлять на выход. Затем мы пропускаем состояние ячейки через ФА tanh (чтобы значения были между -1 и 1) и умножаем его на выходные данные слоя сигмойды, благодаря этому мы отправляем на выход только нужную часть скрытого состояния ячейки (Рисунок 13).

В примере с языковой моделью, поскольку он только что увидел подлежащее, он может захотеть вывести информацию, относящуюся к глаголу, на случай, если это то, что будет дальше. Например, он может выводить, является ли подлежащее единственным или множественным числом, чтобы мы знали, в какую форму следует спрягать глагол.



Рисунок 13 — Output gate

1. КОНСТРУККТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ
   1. Описание поисковых запросов, на которых будет подсчитана метрика

Задача Метрики скорости реакции — измерить качество поисковой

выдачи на запросах, который задавались сразу после наступления новостного события. Таким образом мы можем сформулировать следующие критерии для подходящего запроса (ниже будут даны соответствующие пояснения критериям):

1. С запросом должно быть связано какое-либо новостное событие. Новостное событие должно удовлетворять следующим критерием:
   1. Событие по запросу не является запланированным.
   2. Для объекта запроса произошедшее событие значимо.
2. За день до запроса не должно происходить значимых событий, подходящих под формулировку запроса.
3. Запрос должен быть взят из потока, т.е. являться настоящим существующим запросом, который пользователь задал в какое-либо время в Яндекс Поиск.

Запрос, отвечающий критериям выше будем называть «***Ультрасвежим***». Примеры незапланированных и значимых событий:

* ЧП
* Катаклизм
* Неожиданные заявления
* Происшествия
* Неожиданные смерти

Примеры запланированных событий (запросы, связанные с данными событиями не являются ультрасвежими):

* Телепередача/эфир/стрим
* Спортивный матч
* Премьера фильма / выход новой серии
* Конференции / заседания

Примеры незначимых событий (запросы, связанные с данными событиями не являются ультрасвежими):

* Частые непланируемые события
* Незаметные новости с фронта
* Регулярные заявления

Такие запросы на начальном этапе будут определяться асессорами — сотрудниками, чья работа заключается в выполнении различных заданий для разметки данных. В дальнейшим планировалось обучить модель, которая по временному ряду задавания запроса смогла бы классифицировать запрос на ультрасвежесть.

Для любых самых громких новостных событий характерно следующее поведение: сразу после наступления такого события, количество поисковых запросов, связанных с этим событием, резко возрастает. С какого-то момента ажиотаж вокруг события начинает затухать и динамика запросов по событию падает, ведь люди все меньше и меньше интересуются таким событием.

Проиллюстрируем динамику таких запросов:

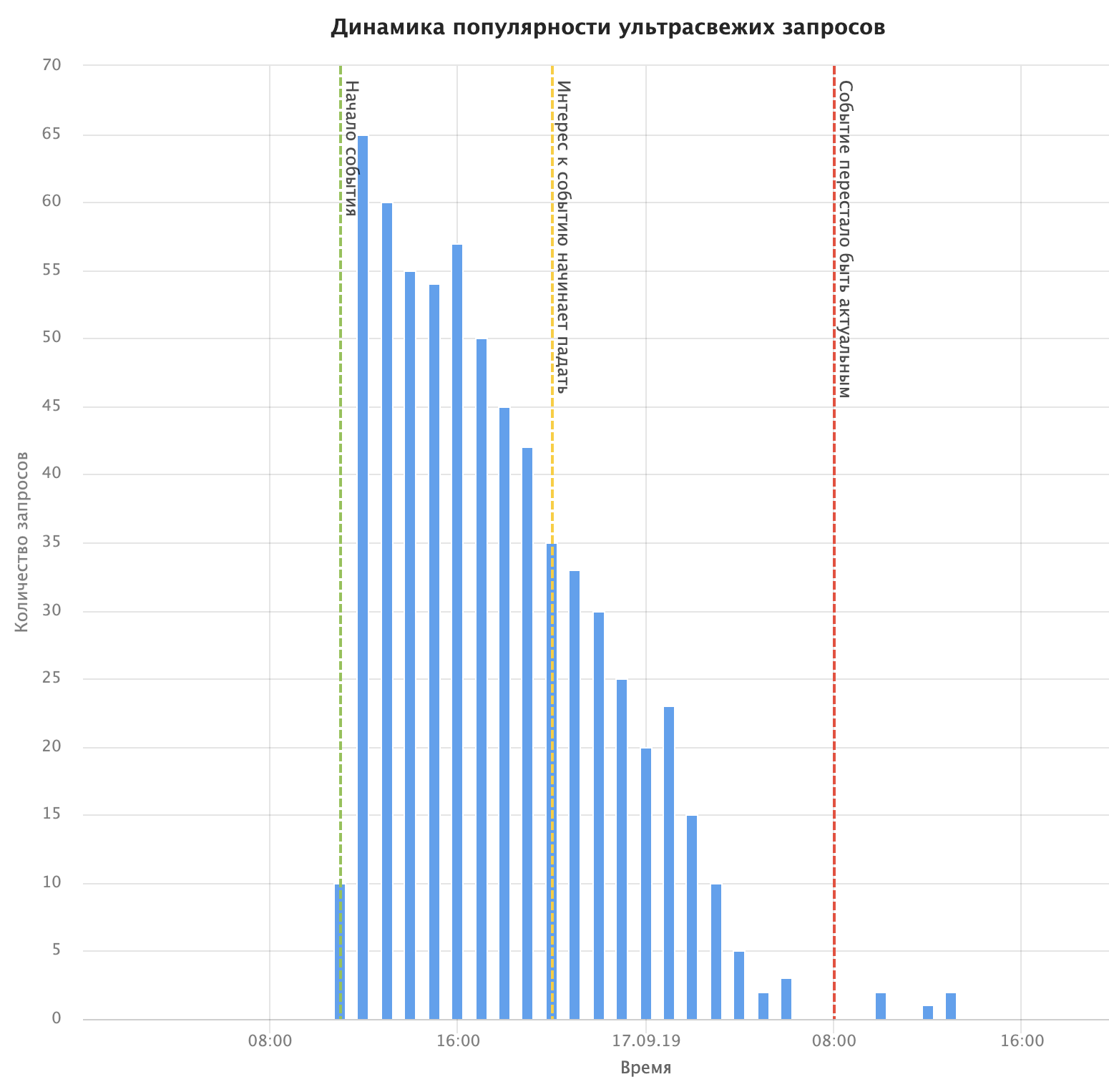


Рисунок 14 — Динамика популярности ультрасвежих запросов

По графику (Рисунок 14) видна отличительная особенность всех ультрасвежих запросов — всплеск популярности и плавный количества поисковых запросов по событию. Чем строже будет соблюдаться такой вид динамики запроса, тем легче в дальнейшем их будет классифицировать модели. Поэтому было решено не рассматривать запланированные события, ведь, поскольку о событии будет известно заранее, люди будут им интересоваться и до его наступления и выделить их среди потока с необходимой точностью не получится. Дополнительным условием для запроса является значимость события. Если событие незначимо, то оно не вызовет резонанс общественности, характерный всплеск не будет наблюдаться и моделе будет сложнее такой запрос верно классифицировать. Чтобы наблюдался резкий всплеск запроса также необходимо, чтобы за день до по данному запросу не было релевантных событий.

По нашим предположениям, если учтены нюансы, о которых сказано выше, то модель сможет с достаточной точностью классифицировать запросы на ультрасвежесть и тогда затраты на необходимый объем разметки асессорами будут укладываться в бюджет компании на соответствующее полугодие.

* 1. Запуск разметки на ультрасвежесть

После анализа природы необходимых запросов и формализованного описания термина ультрасвежести необходимо набрать необходимое количество асессоров и составить проект разметки в системе Yang.

Yang — краудсорсинговый проект для разметки большого количества данных, которые затем, как правило, используются для машинного обучения.

Асессоры должны были ознакомиться с инструкцией, пройти составленные мной обучающие и экзаменационные задания. Асессоры успешно справившееся с экзаменом допускаются до разметки реальных данных (Рисунок 15).

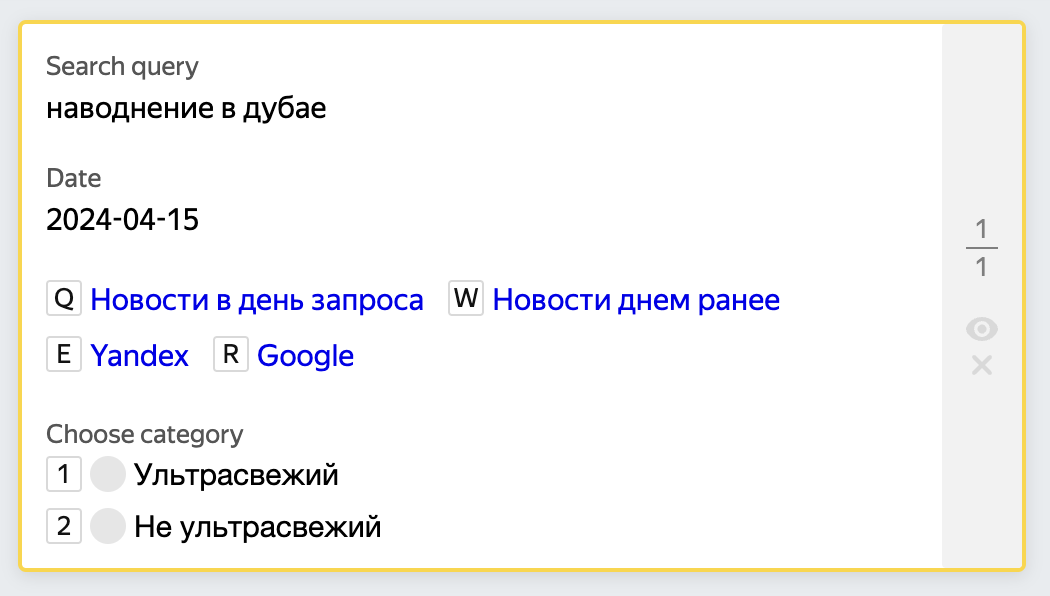


Рисунок 15 — Интерфейс асессорской разметки

* 1. Ансамбль для классификации ультрасвежих запросов

После получения необходимого количества разметки необходимо понять,

как на этом датасете обучить модель для классификации ультрасвежих запросов.

В нашем распоряжении были логи самого поиска и вектор признаков запроса, который автоматически считается Яндекс Поиском. Для классификации необходимо было обработать динамику запроса в соответствующий день (пример Рисунок 14), а также использовать уже имеющийся посчитанный вектор признаков, который описывает смысл запроса.

Было решено использовать LSTM для обработки временных рядов динамики запроса, т.к. эта модель хорошо себя показывает на работе с последовательностями, и ее архитектура не такая сложная, как, например, BERT, чтобы быстро переобучиться на нашем небольшом датасете (асессорский ресурс — небесплатный).

Для финальной классификации поверх LSTM было решено использовать CatBoost т.к. эта модель, как правило, показывает самые высокие результаты на табличных данных. В нашем случае данными для классификации будут признаковое описание запроса и получившееся предсказания LSTM.

Итоговый пайплайн можно проиллюстрировать следующим образом (Рисунок 16).

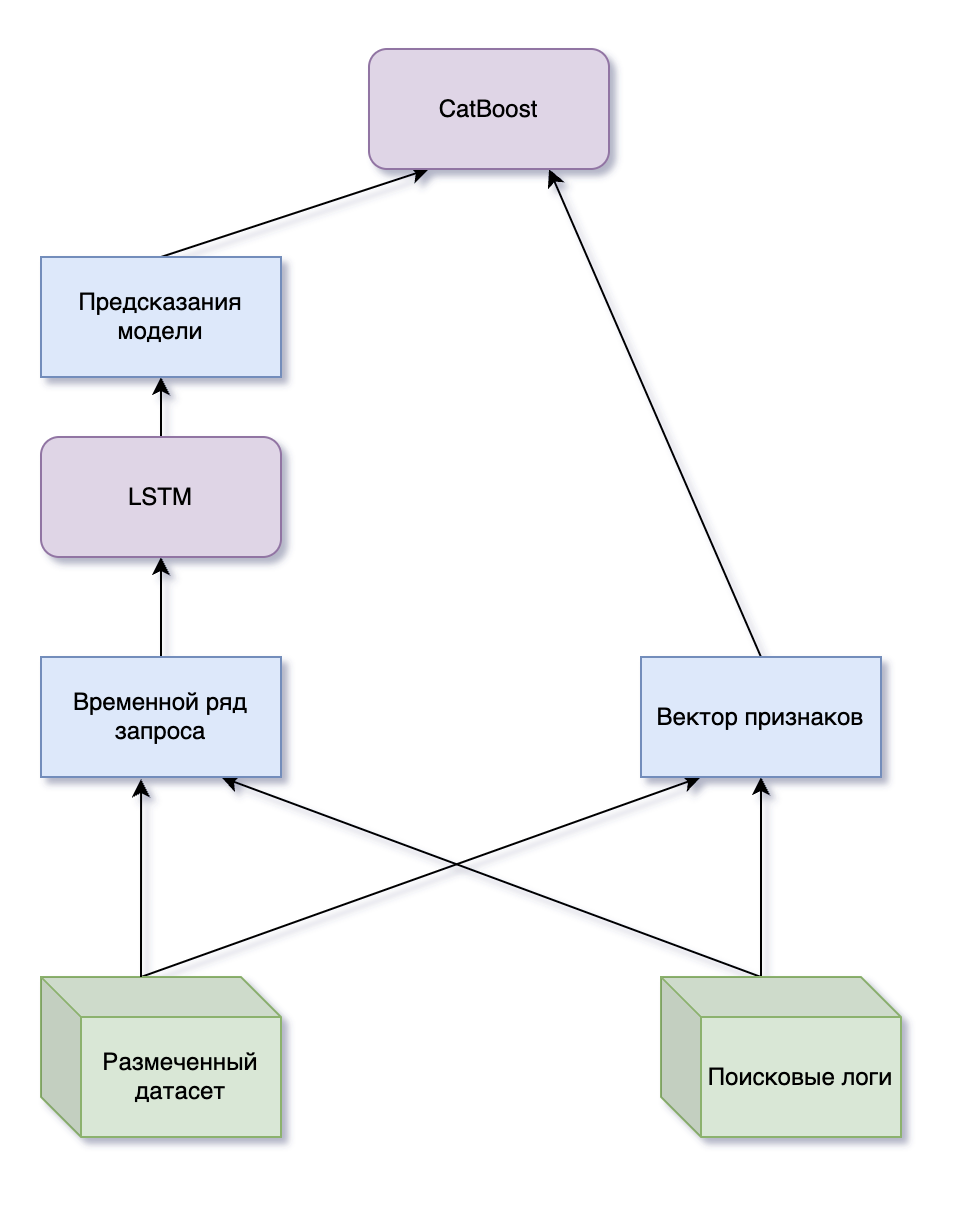


Рисунок 16 — Пайплайн классификации на ультрасвежесть

* 1. Обучение LSTM

По временному ряду необходимо предсказать вероятность отнесения данного запроса к ультрасвежим.

Пример временного ряда ультрасвежего запроса:

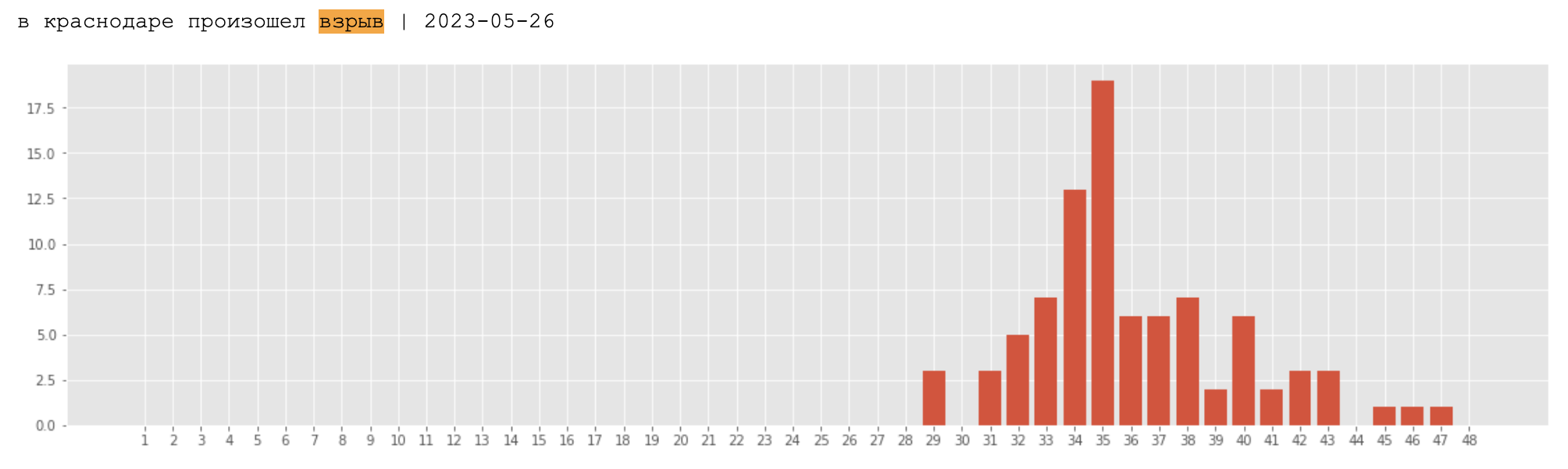


Рисунок 17 — временной ряд ультрасвежего запроса

Пример временного ряда новостного, но не ультрасвежего запроса про заранее спланированное спортивное событие (видим отсутствие всплеска):

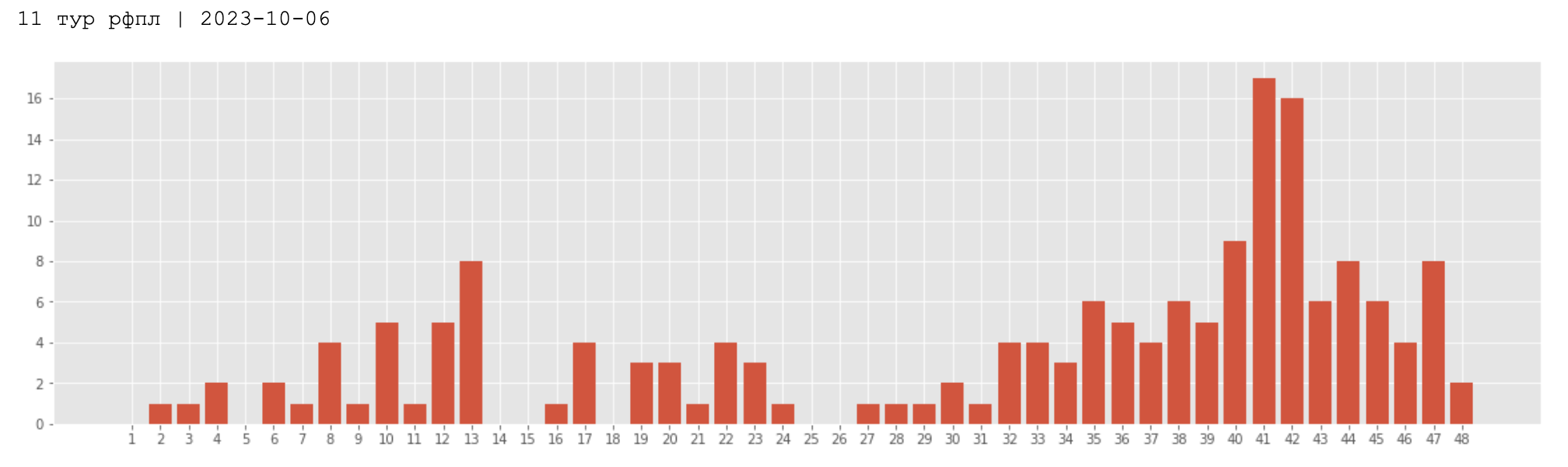


Рисунок 18 — временной ряд запроса, связанного с заранее спланированным событием

Такой запрос не будет являться ультрасвежем по нашему определению, так как о событии заранее известно.

Нормализовав такой временной ряд можно попробовать обучить первую модель нашего ансамбля для классификации запроса — LSTM. Ниже представлен листинг кода основного класса модели.

class LSTMClassifier(nn.Module):

"""Implementation of LSTM-based time-series classifier."""

def \_\_init\_\_(self, input\_dim, hidden\_dim, layer\_dim, output\_dim):

super().\_\_init\_\_()

self.hidden\_dim = hidden\_dim

self.layer\_dim = layer\_dim

self.rnn = nn.LSTM(input\_dim, hidden\_dim, layer\_dim, batch\_first=True)

self.fc = nn.Linear(hidden\_dim, output\_dim)

self.batch\_size = None

self.hidden = None

def forward(self, x):

h0, c0 = self.init\_hidden(x)

out, (hn, cn) = self.rnn(x, (h0, c0))

out = self.fc(out[:, -1, :])

return out

def init\_hidden(self, x):

h0 = torch.zeros(self.layer\_dim, x.size(0), self.hidden\_dim)

c0 = torch.zeros(self.layer\_dim, x.size(0), self.hidden\_dim)

return [t.cuda() for t in (h0, c0)]

В качестве оптимизитора градиентного спуска использовался Adam т.к. он обеспечивает самую быструю сходимость и умеет преодолевать точки локального минимума функции потерь. Поверх выхода нейронной сети применялась функции активации сигмойды т.к. решаем задачу бинарной классификации.

Основные параметры модели:

* Размер скрытого полносвязного слоя идущего сразу после LSTM — 256 (hidden\_dim в коде).
* Количество эпох — 100
* Learning rate — 0.0005

При обучении использовался шедулер, засчет которого можно было циклически изменять learning\_rate, чтобы не застревать в точках локального минимума и с наилучшем приближением придти к глобальному экстремуму. Ниже представлен график коэффицента, на который с течением времени будет убножаться learning rate:

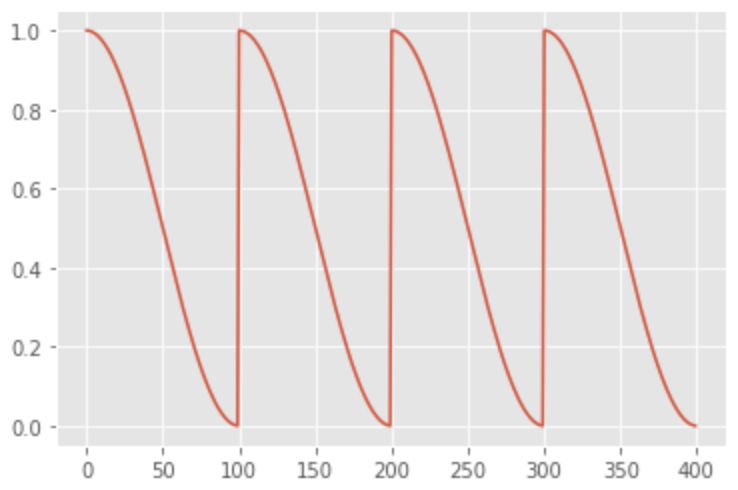


Рисунок 19 — коэффициент умножения скорости обучения

Итоговые метрики при пороге 0.403 при бейзлайне (предсказание ультрасвежести на всех запросах) precision = 0.05, recall = 1:

|  |  |  |
| --- | --- | --- |
| Метрика | Precision | Recall |
| Значение | 0.14 | 0.96 |

Стоит отметить, что бизнес требованием к данной задаче являлась высокая полнота классификации запросов. Идея в том, чтобы как можно более честно брать ультрасвежие запросы из потока, без смещений на модели. В случае низкого recall часть запросов бы терялось, потому что их не различила бы модель, а возможно именно на этих запросах у поисковой системы низкое качество.

Учитывая, что раньше лишь 0.05 запросов по разметке было ультрасвежими делаем вывод, что засчет данной модели необходимо в 3 раза меньше асессорской разметки, чтобы набрать такое же количество ультрасвежих запросов за определенный день. Поэтому обучение модели можно считать успешным.

* 1. Обучение градиентного бустинга

В качестве фреймворка градиентного бустинга был выбран CatBoost, на данный момент это лучший технологический фреймворк, который позволяет гибко обучить модель и настроить визуализацию обучения, необходимую тебе.

Датасет на котором обучалась модель содержал разные нейросетевые признаки запроса такие как: вероятность отнесения запроса, к новостным или спортивным, является ли запрос медицинским и т.д. Всего более 300 колонок с разными текстовыми признаками. Временной ряд каждого запроса был обработан LSTM, ее выход был добавлен в качестве дополнительного признака.

За отложенное нам время нам удалось разметить почти 20 тысяч разных запросов. В датасете была всего 1 тысяча ультрасвежих запросов, поэтому необходима была стратификация по предсказываемому параметру перед делением данных на тренировочную и тестовую части. Несмотря на стратификацию обучение катбуста и его финальные метрики очень сильно зависили от рандома разбиения датасета на тестовую и тренировочную части. Становилось ясно, что необходимо кросс-валидация — метод оценки производительности модели, состоящий в том, чтобы разбить обучающую выборку на несколько фолдов (частей), проверить качество модели на каждом фолде, используя остальные фолды для обучения, и усреднить результаты. Этот метод позволяет получить более точную оценку производительности модели, т.к. использует все доступные данные для обучения и валидации.

Т.к. выборка несбалансирована, метрика точности классификации не подходит т.к. ее значение будет не информативным, ведь точность бейзлайна (предсказание константы 0) достигало бы уже 95% т.к. в исходном датасете всего 5% ультрасвежих запросов.

Продуктово нам было необходимо, чтобы модель как можно меньше пропускала ультрасвежих запросов для более точного и объективного подсчета метрики, поэтому было принято решение максимизировать precision при recall >= 0.95.

Ниже представлен график обучение градиентного бустинга на каждом из фолдов кросс-валидации (всего их было 50).

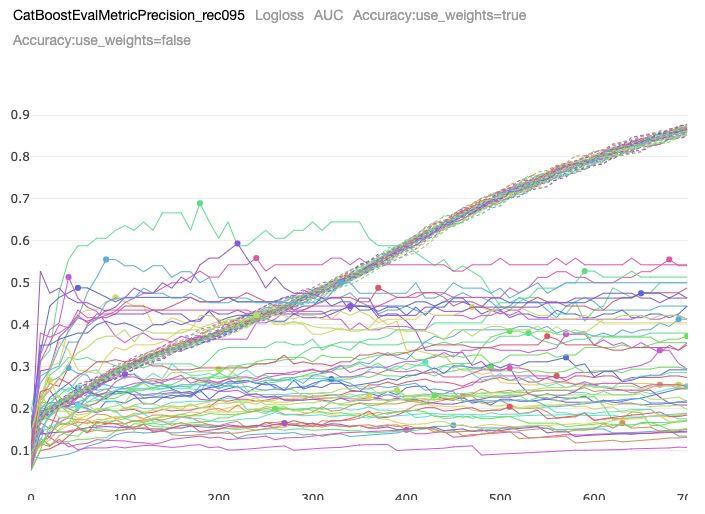


Рисунок 20 — график precision модели при recall >= 0.95 на каждом фолде кросс-валидации

Такие графики генерирует фреймворк CatBoost при указании необходмой конфигурации обучения. По оси Ox показывается количество обученных деревьев по оси Oy значение выделенной жирным метрики (на Рисунке 20 выбрана самописная метрика — значение precision при кecoll не меньше 0.95). Сплошной линией показывается значение метрик на тестовом датасете, пунктирной — на тренировочном. Точкой отмечено количество деревьев в градиентном бустинге при котором значение метрик максимальной на данном фолде. Как только значение метрик на тестовом множестве перестает падать градиентный бустинг переобучается и обучение следует остановить, но на каждом фолде необходимо обучить разное количество деревьев, прежде чем модель начнет переобучаться, поэтому было зафиксировано, что на каждом фолде мы последовательно обучаем 700 деревьев.

Ниже представлен график метрики LogLoss, где каждый цвет также определяет определенный фолд кросс-валидации, обозначения, описанные выше для предыдущего графика также верны:

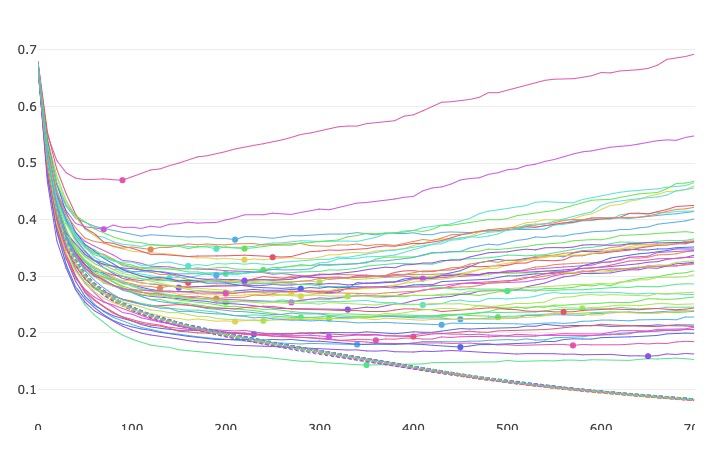


Рисунок 21 — значение LogLoss на каждом фолде кросс-валидации

Чтобы выбрать итоговое количество деревьев необходимо выбрать количество итераций обучения деревьев, при котором среднее значение целевой метрики precision (Рисунок 20) больше усредненного значения метрики на любом другом количестве итераций обучения деревьев.

Итоговые метрики градиентного бустинга:

|  |  |  |
| --- | --- | --- |
| Метрика | Precision | Recall |
| Значение | 0.3 | 0.95 |

Взятое количество деревьев — 410, глубина каждого дерева — 5. Порог при котором посчитаны метрики — 0.29 (градиентный бустинг предсказывает вероятность класса ультрасвежести для запроса).

Сравнивая разные подходы предсказания были получены следующие метрики (recall в каждом случае был не меньше 0.95):

1. LSTM, обученная на временном ряде запроса показывала precision = 0.15.
2. CatBoost, обученный на нейросетевых признаках, используемых в продакшене Яндекс Поиска показал precision = 0.25.
3. Если добавить предсказания LSTM, как дополнительный признак и обучить CatBoost, то precision = 0.3, что является наилучшим показателем.
   1. Итоговый пайплайн подсчета метрик

После того, как мы научились определять запросы, на которых хотим замерять качество необходимо реализовать итоговый процесс подсчета метрики. Для каждого ультрасвежего запроса необходимо определить момент начала этого события, точнее, момент, когда по данному событию был создан первый релевантный документ. Начиная с этого момента, мы будем считать, что миру стало известно о данном событии, а значит, если это событие громкое, то в скором времени ожидается пик запросов по данному событию, на таких запросах мы хотим померить качество.

Чтобы определить, когда был создан первый релевантный документ к этому запросу необходимо взять из базы индексированных страниц поиска страницы, созданные в день запроса, а далее в автоматическом режиме определить, какие из этих страниц релевантны запросу. Эту задачу мы реализовали с помощью архитектуры BERT нескольких типов: запросный BERT, и документный BERT. На выходе каждой модели мы получали вектор признаков, описывающий запрос и страницу соответственно. Значение скалярного произведения будет являться мерой схожести запроса и документа. Подобрав порог скалярного произведения можно автоматически отбирать страницы, релевантные запросу.

Страницы, релевантные запросу отправляются на запросно-документную асессорскую разметку (Рисунок 22), где асессоры отвечают на вопрос «Содержит ли данная страница актуальную новость по указанному запросу?».

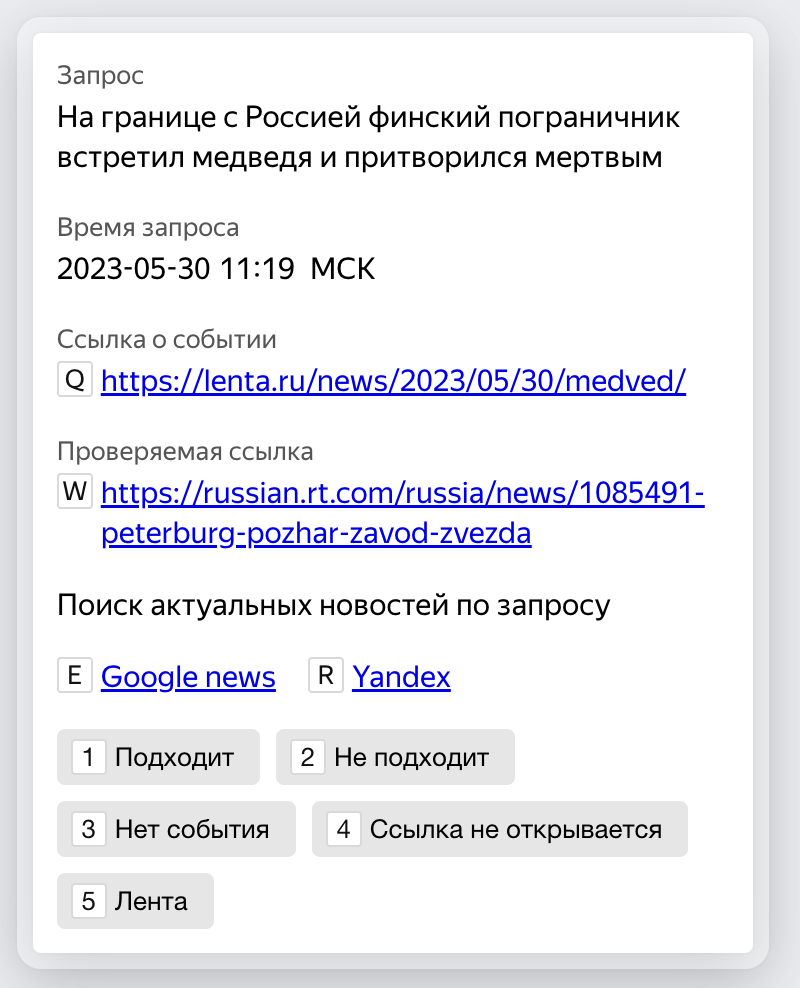


Рисунок 22 — асессорская разметка на соответствие страницы ультрасвежему запросу