

Введение в информационный поиск

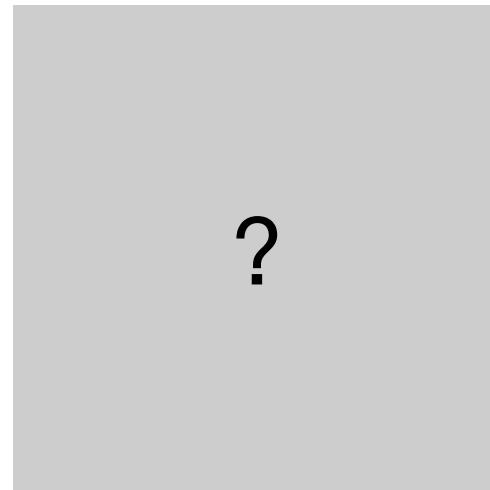
Андрсов Дмитрий, 09.02.2026, AI Masters

План лекции

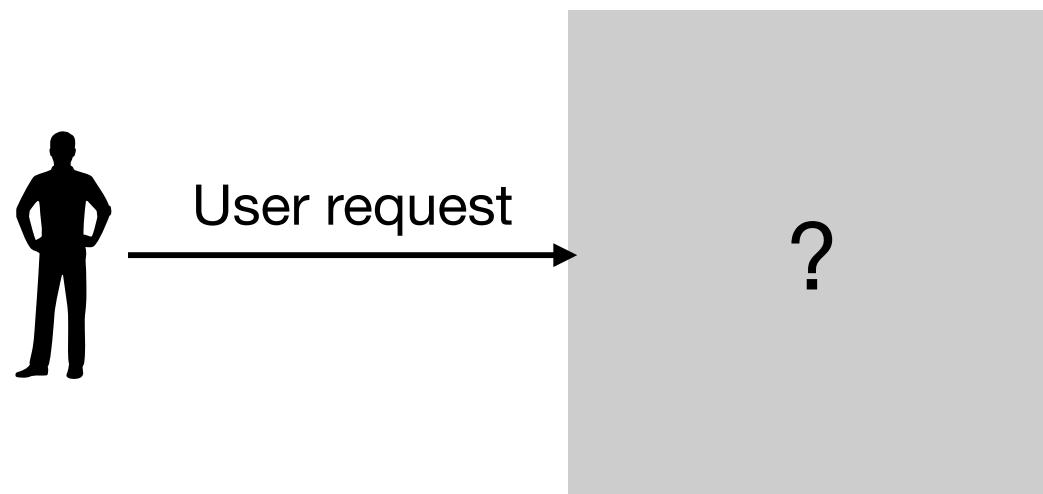
- Задача информационного поиска
- Обработка текста
- Обратный индекс
- Представление поискового запроса

Введение

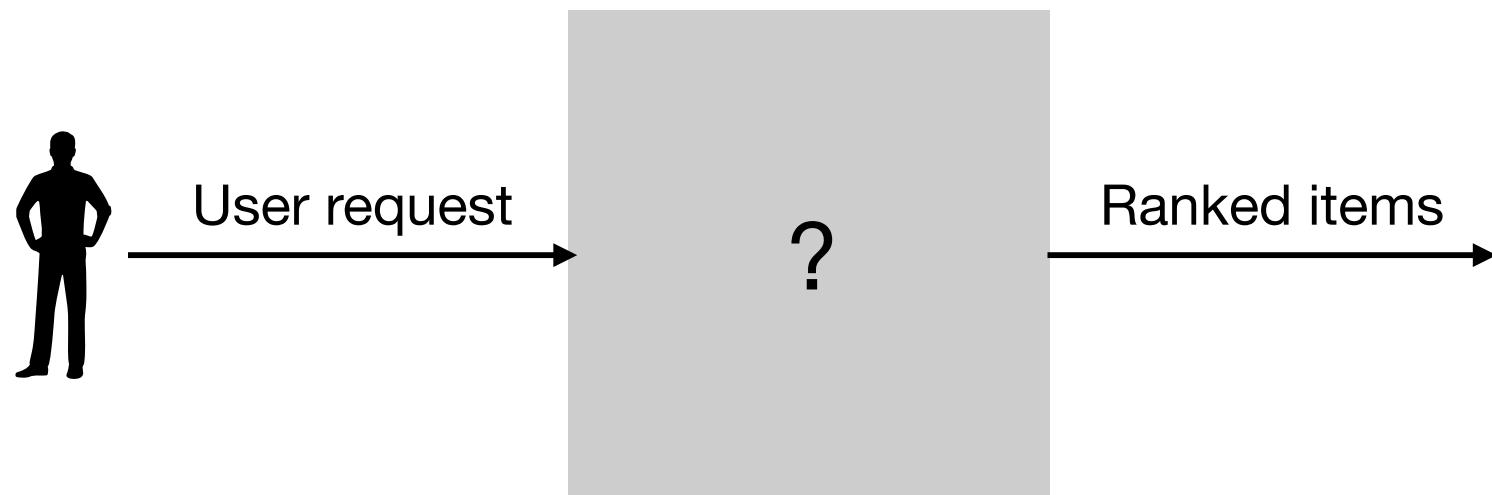
Ранжирующая система



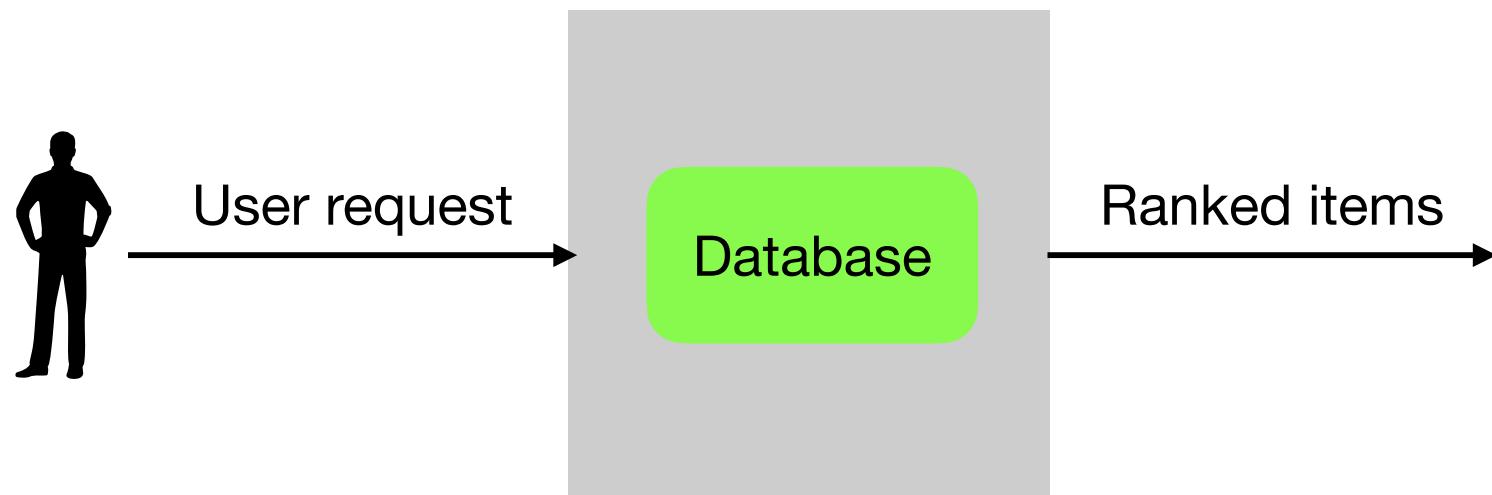
Ранжирующая система



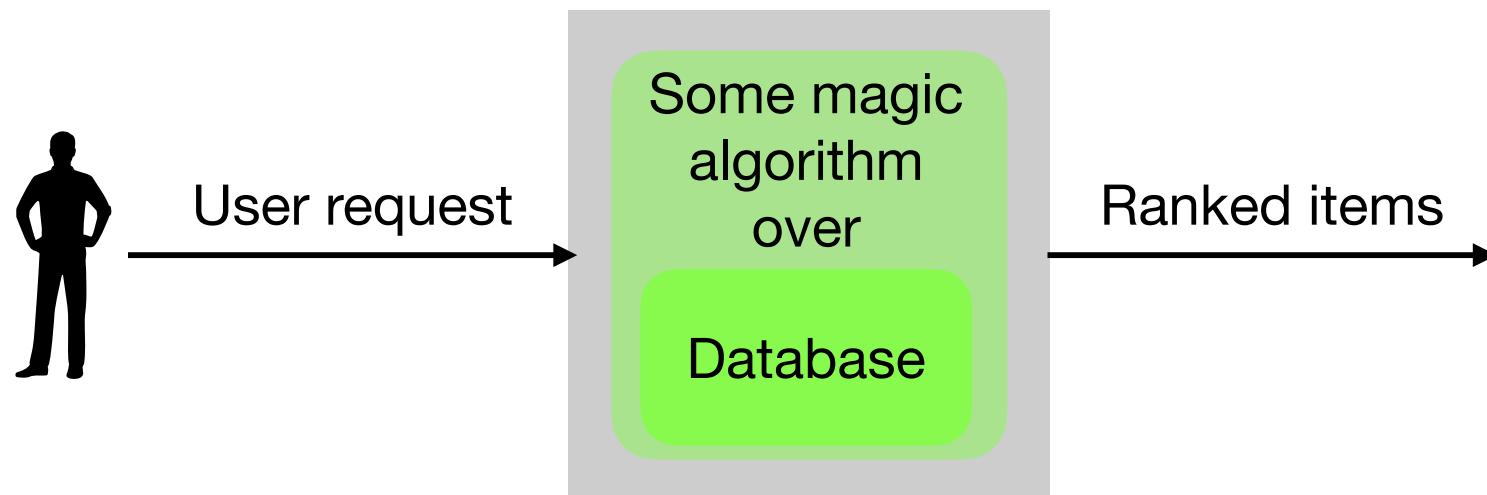
Ранжирующая система



Ранжирующая система



Ранжирующая система



Что такое информационный поиск?

Информация (Information) — совокупность фактов и знаний (неструктурированной информации), из которых можно сделать какой-то вывод — коллекция документов (корпус).

Поиск (Retrieval) — процесс извлечения чего-либо откуда-либо.

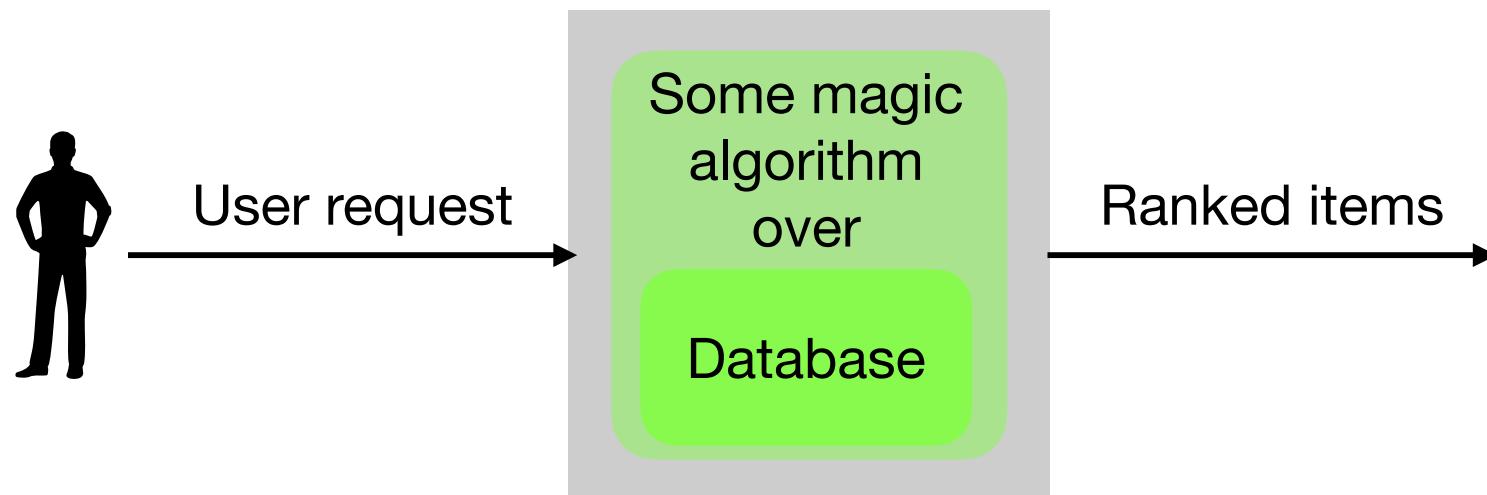
Информационный поиск (Information Retrieval) — процесс извлечения документов, состоящих из неструктурированной информации, удовлетворяющих потребностям пользователя, из корпуса.

Что такое информационный поиск?

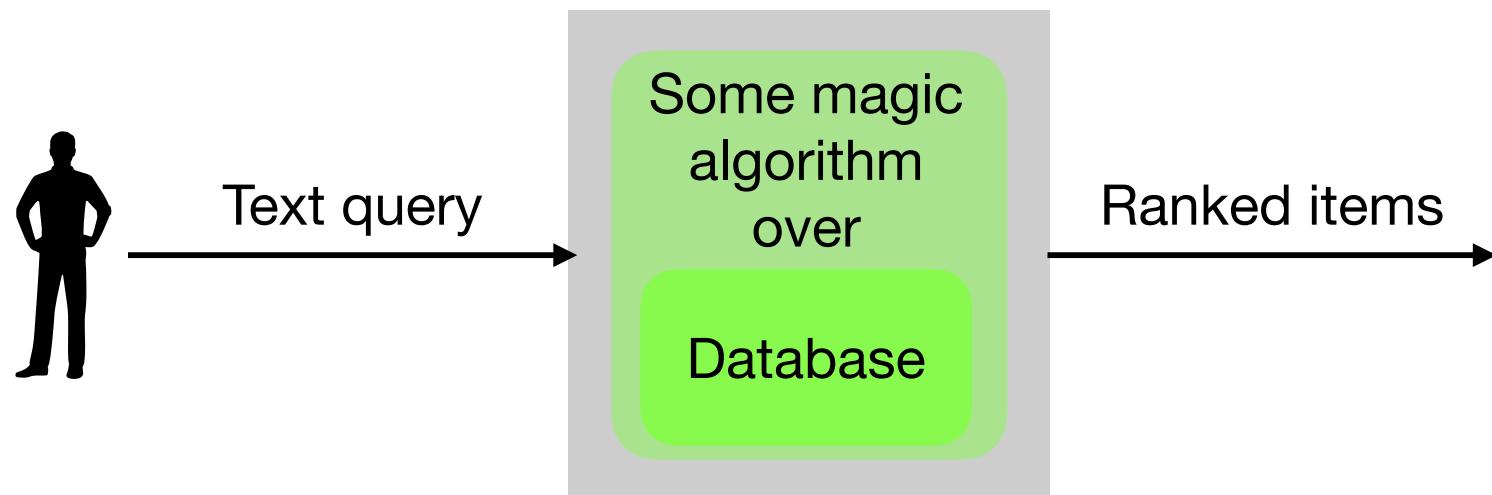
Информационный поиск — процесс извлечения документов, состоящих из **неструктурированной информации**, удовлетворяющих **потребностям пользователя**, из коллекции документов.

- **Неструктурированная информация:** нет определенной структуры (например, текст);
 - Типы информации в документах: текст, аудио, видео, изображения и т.д.
- **Потребности пользователя:** поисковый запрос.

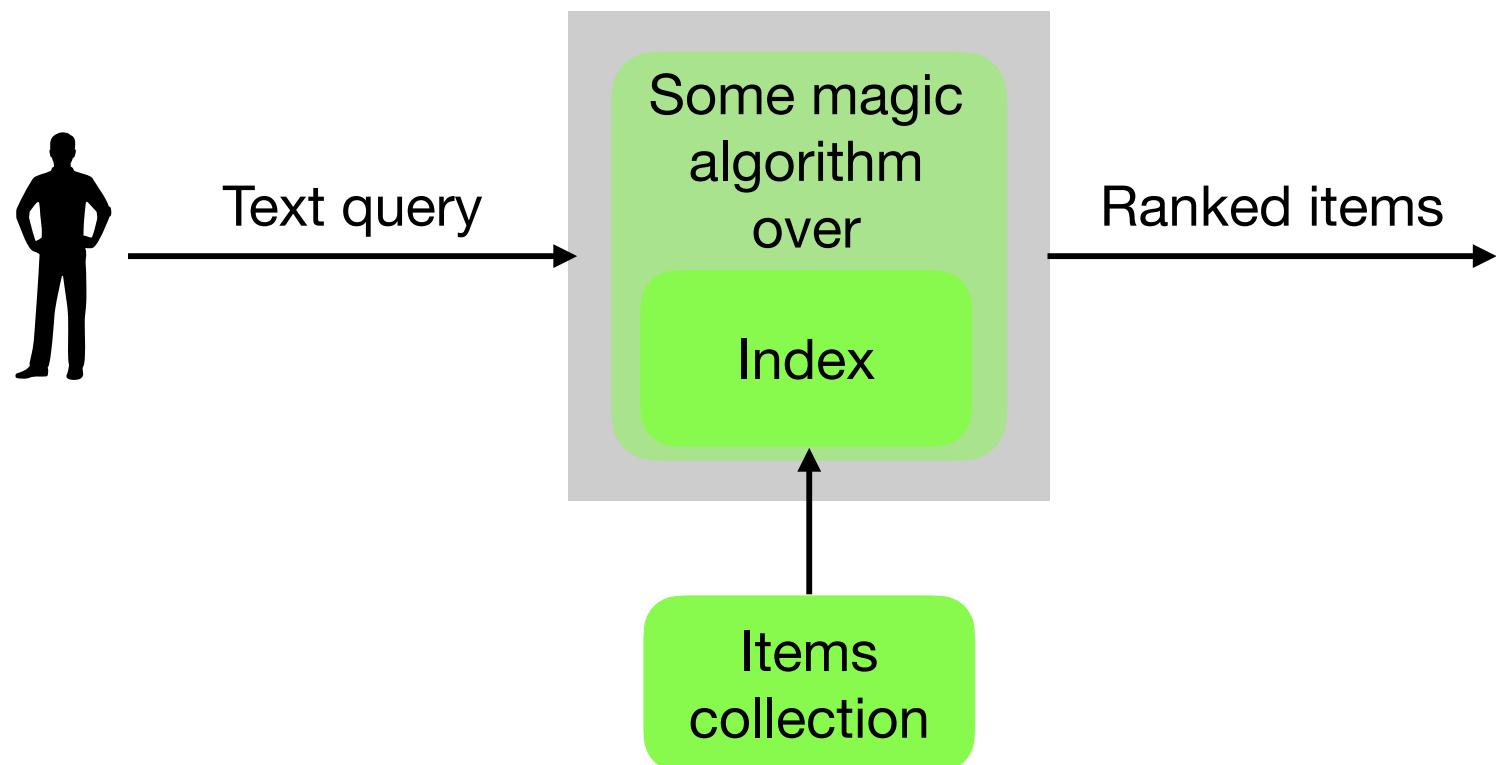
Ранжирующая система



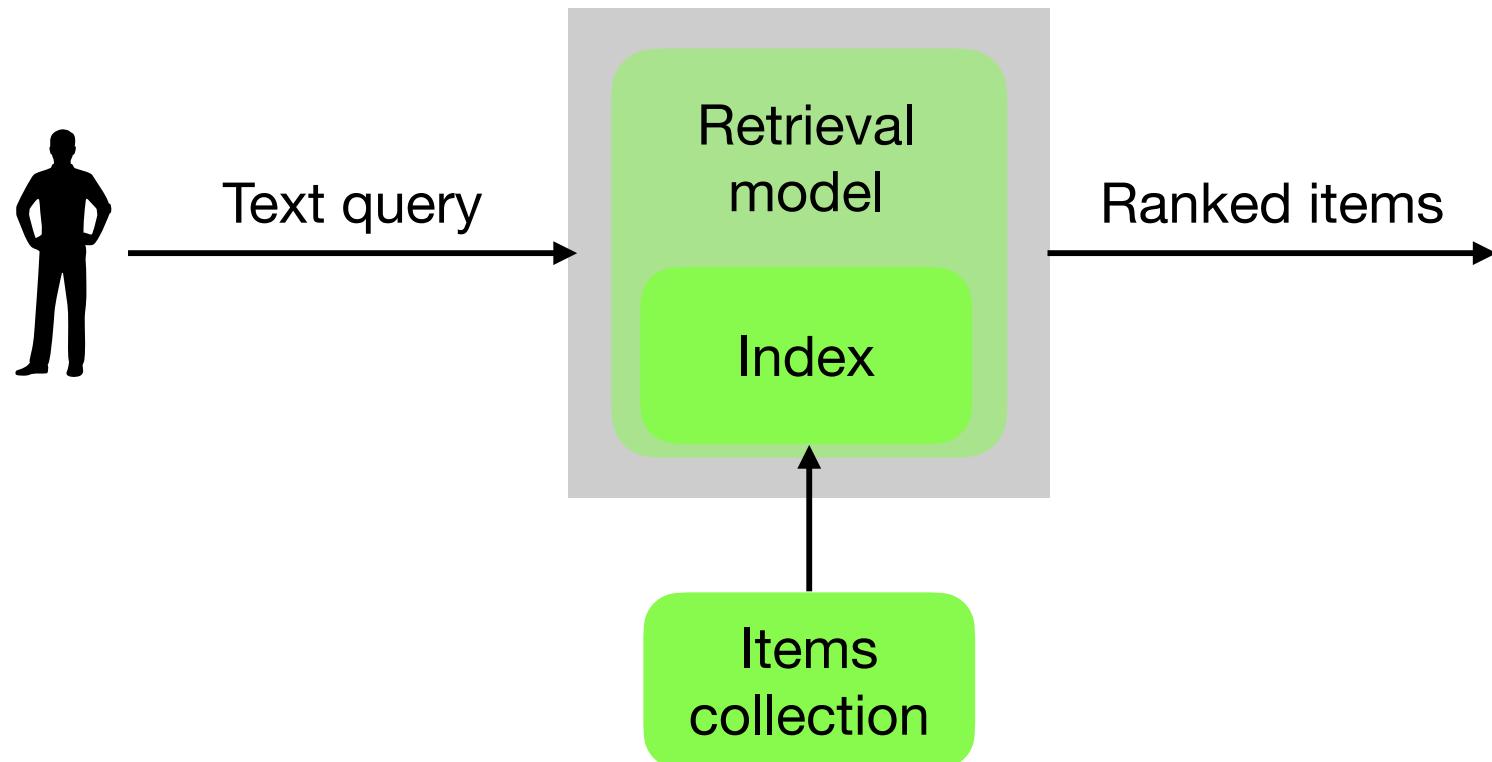
Поисковая система



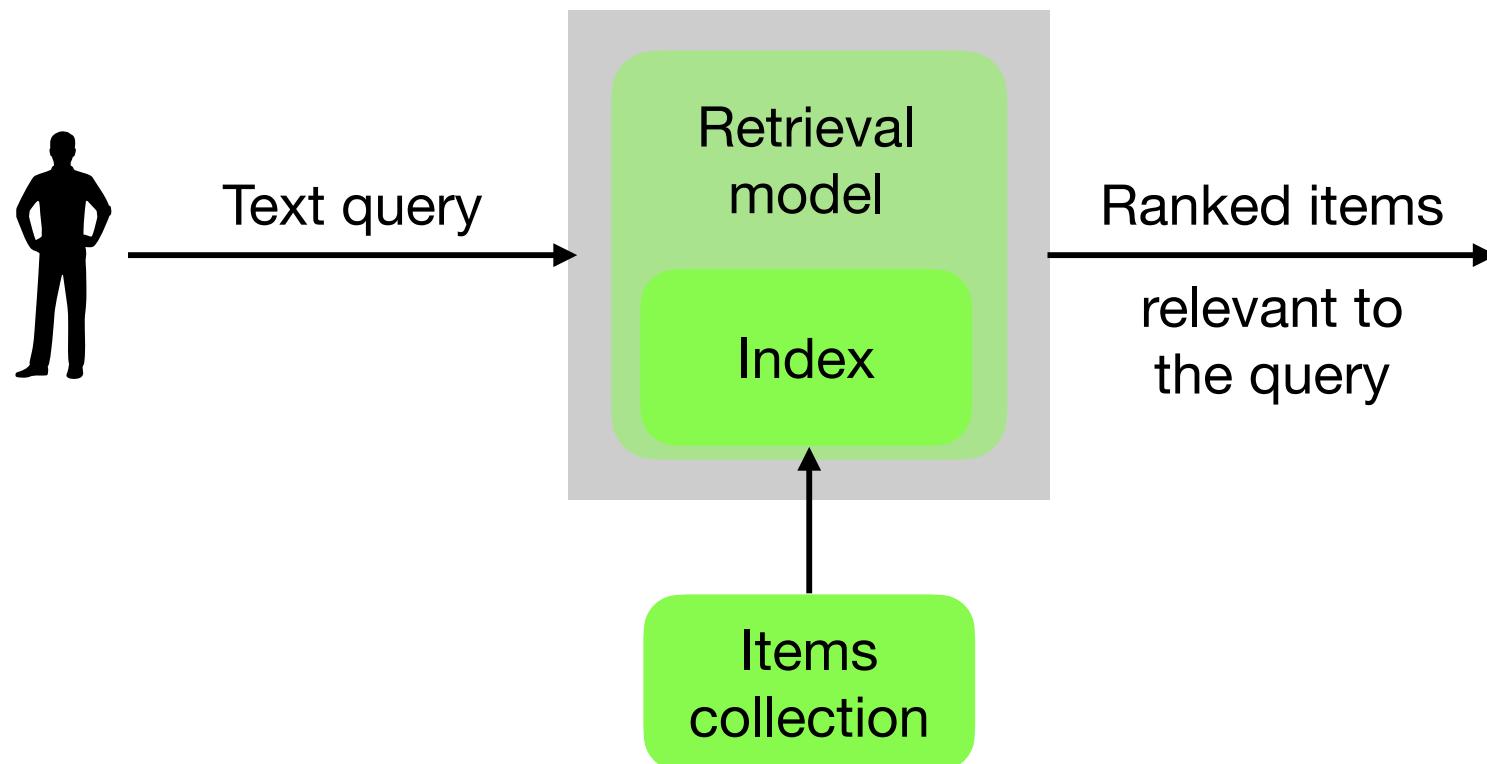
Поисковая система



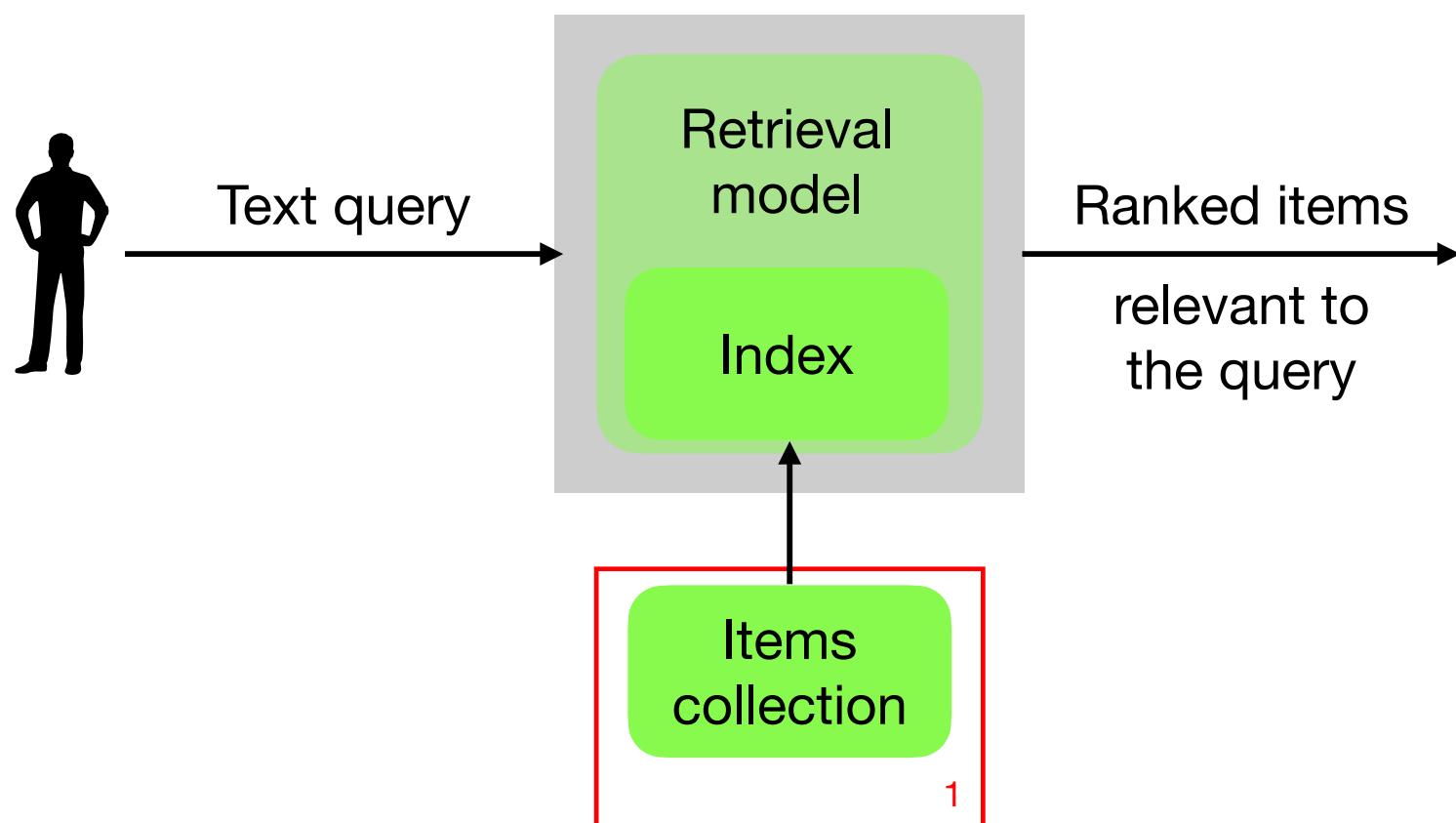
Поисковая система



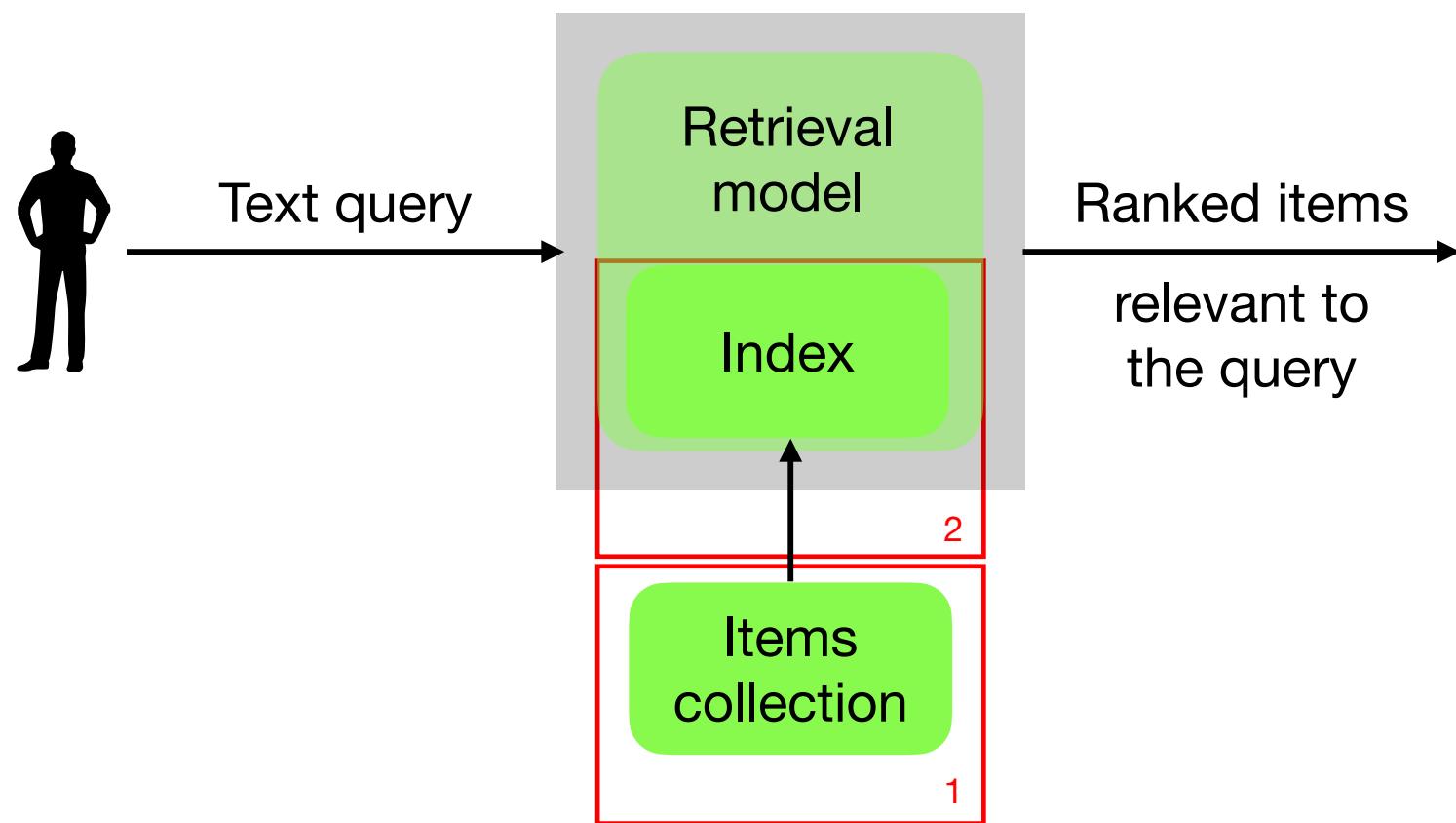
Поисковая система



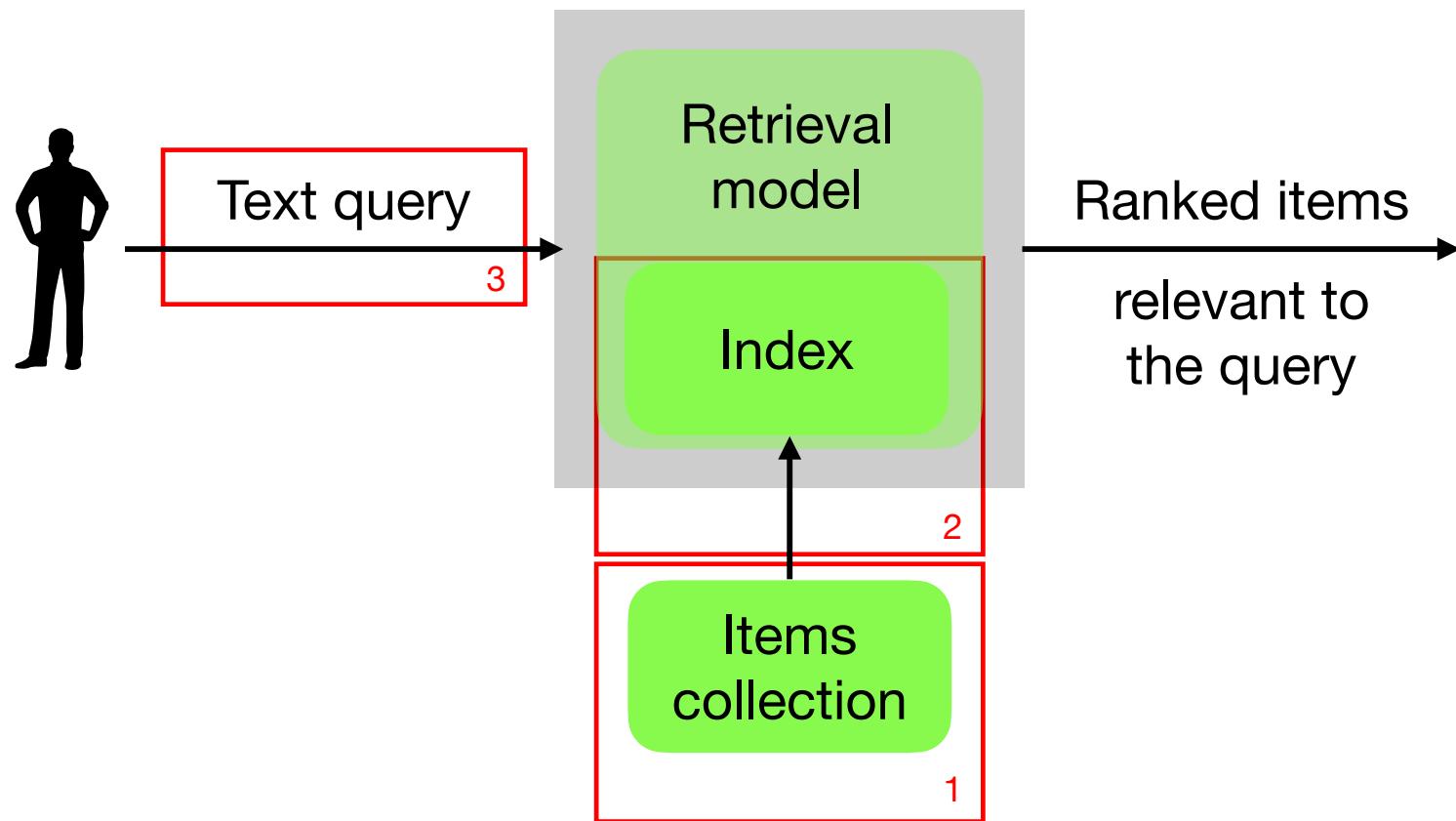
Поисковая система



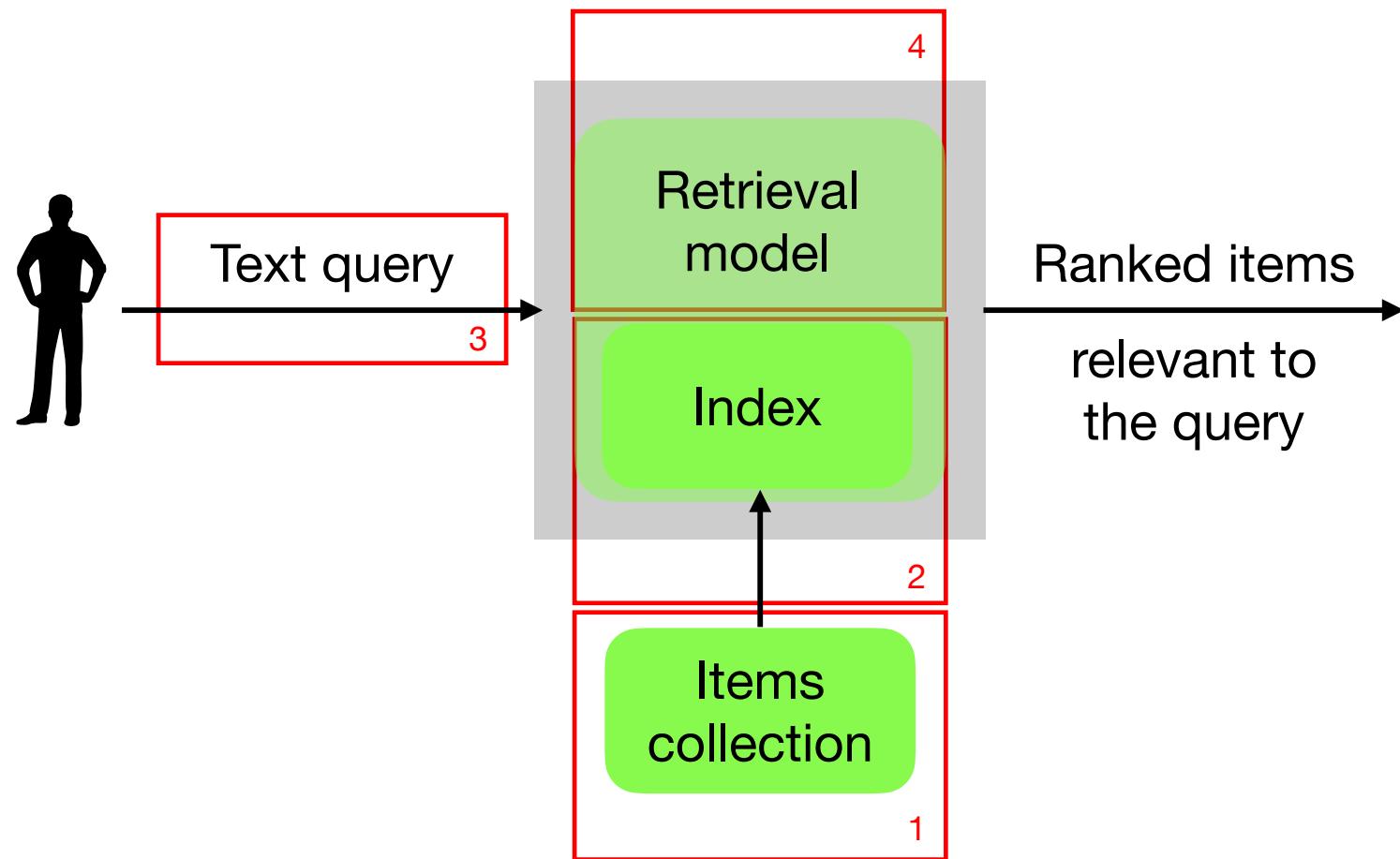
Поисковая система



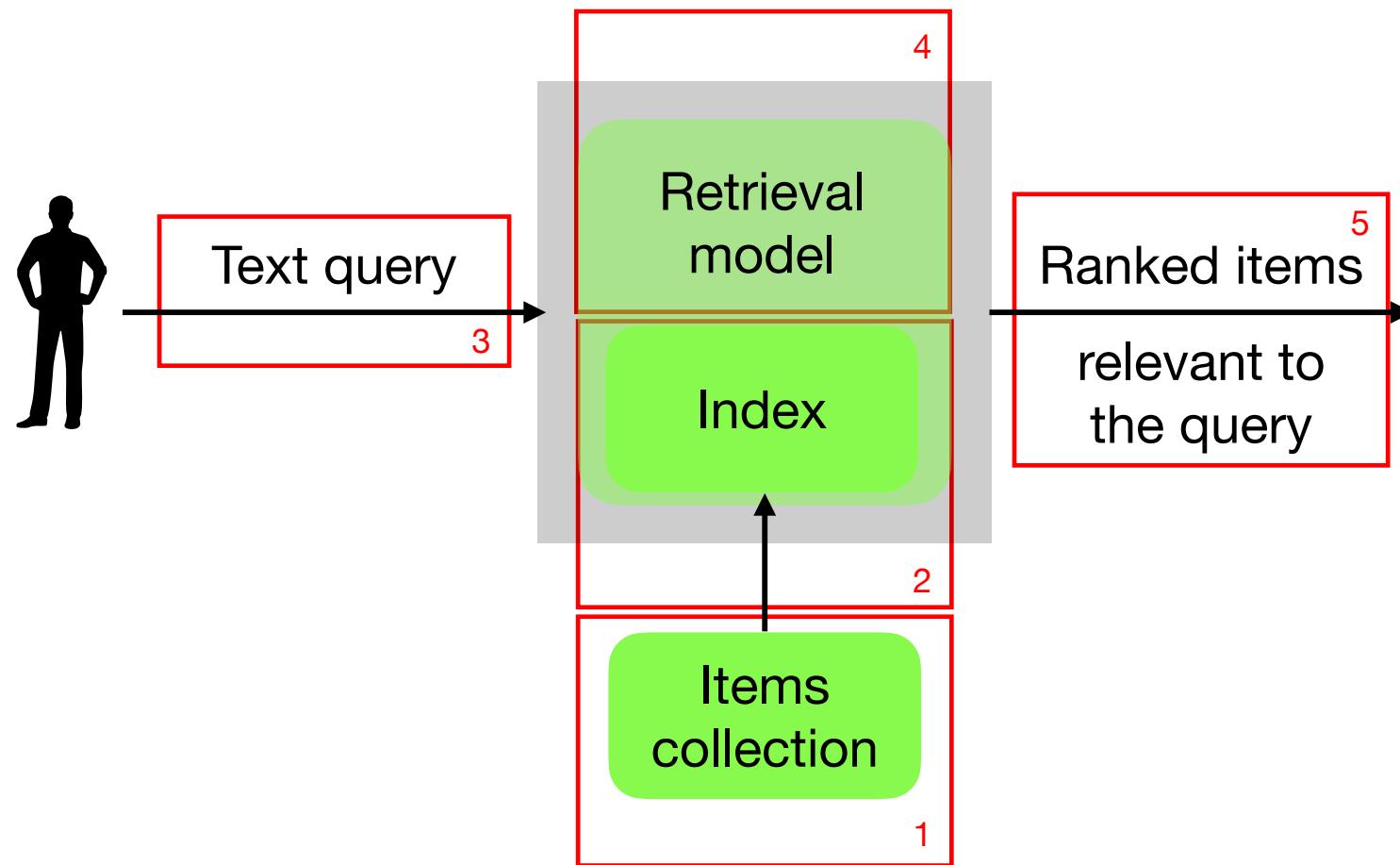
Поисковая система



Поисковая система



Поисковая система



Поисковая система

- Обработка документов коллекции;
- Построение поискового индекса;
- Обработка запроса пользователя;
- Алгоритм поиска документов индекса, соответствующих поисковому запросу пользователя;
- Оценка качества поиска.

Примеры поисковых систем

Примеры поисковых систем

Web Search

Google search results for "что такое information retrieval".

Search bar: что такое information retrieval

Filter: Все

Result 1: Обзор от ИИ
Information Retrieval (информационный поиск, ИИ) — это область компьютерных наук, занимающаяся поиском неструктурированной информации (обычно текста, документов, изображений), соответствующей информационным потребностям пользователя, выраженным в запросе. Это процесс поиска, индексирования, ранжирования и выдачи наиболее релевантных материалов из больших коллекций данных.

Result 2: Основные аспекты Information Retrieval:

- Цель: Найти данные, соответствующие смыслу запроса, а не просто точное совпадение слов.

Result 3: Информационный поиск - Википедия
Информационный поиск (англ. information retrieval) — процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности, ...

Result 4: Информационный поиск ? - IBM
Поиск информации (IR) — это широкая область компьютерных и информационных...

Result 5: datascientist.one
Information Retrieval - информационный поиск - Data Science
5 апр. 2016 г. — Информационный поиск (information retrieval) — процесс поиска неструктурированной документальной информации, удовлетворяющей информационные ...

Яндекс search results for "что такое information retrieval".

Search bar: что такое information retrieval

Section: поиск с алисой

Result 1: Алиса AI
Быстрый ответ, возможны неточности

Result 2: Процесс поиска

Result 3: Information retrieval — информационный поиск. Это процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности, и наука об этом поиске.
ru.ruwiki.ru ru.wikipedia.org*

Result 4: Цель информационного поиска — помочь пользователю найти необходимую информацию быстро и эффективно. glean.com

Result 5: ru.ruwiki.ru ru.wikipedia.org* gpntb.ru tpc.ispras.ru

Result 6: W Information retrieval - Wikipedia
en.wikipedia.org › Information retrieval
In information retrieval, a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevance.
PKH: иностранный владелец ресурса нарушает закон РФ

Result 7: W Информационный поиск — Википедия
ru.wikipedia.org › Информационный поиск
Информационный поиск (англ. information retrieval) — процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности, и наука об этом поиске.
PKH: иностранный владелец ресурса нарушает закон РФ

Примеры поисковых систем

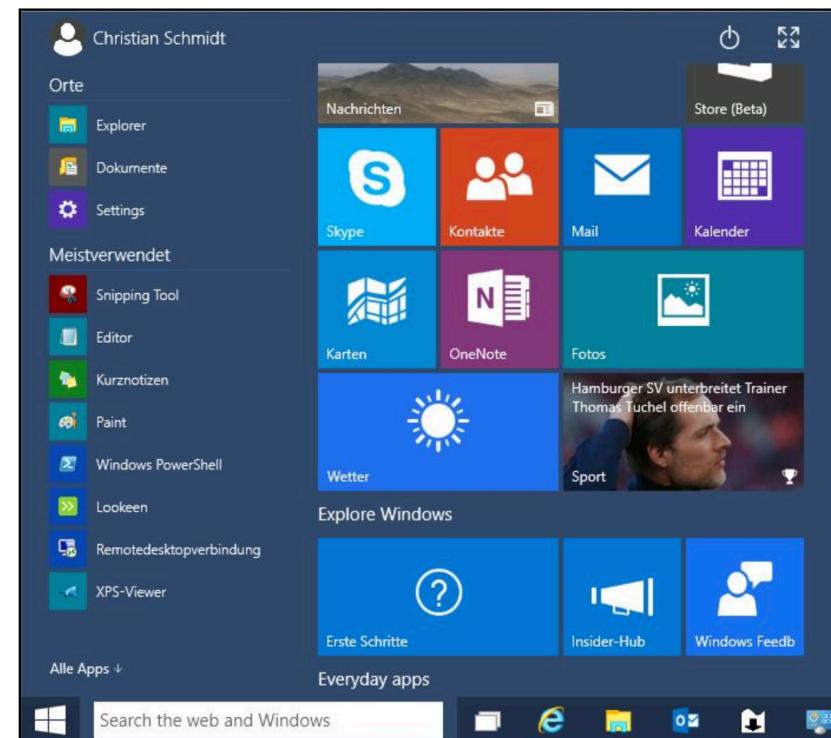
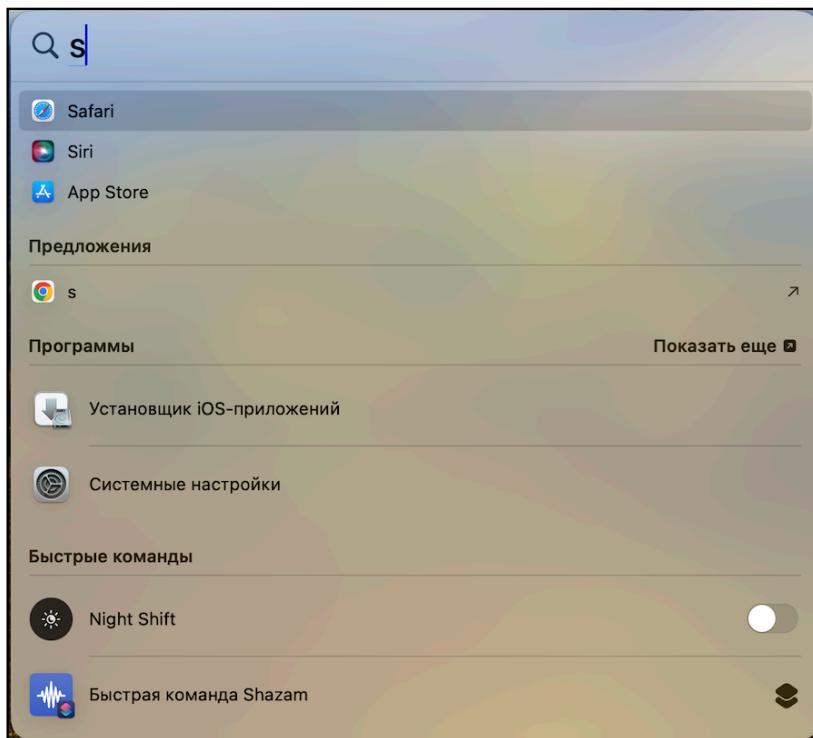
Product Search

The screenshot shows the Ozon search results for men's jackets. The interface includes a header with the Ozon logo, a search bar, and navigation links like 'Каталог' and 'Вход'. Below the search bar, there are filters for categories like 'Тактическая куртка', 'Куртка multicam', and 'Куртка мужская демисезонная больших размеров...'. A sidebar on the left contains filters for 'Категория', 'Распродажа', 'Сроки доставки', 'Цена', 'Рассрочка', 'Бренд', and 'Сообщество'. The main content area displays a grid of four men's jackets, each with a price, discount percentage, availability status ('1шт осталось'), brand name ('PORTNOVERS'), and rating ('4.9'). Each item has a 'Распродажа' (Sale) badge.

The screenshot shows the Wildberries search results for men's jackets. The interface features a header with the Wildberries logo and a search bar. Below the search bar, there are various filters and sorting options. The main content area displays a grid of men's jackets, each with a price, discount percentage, availability status ('1шт осталось'), brand name ('PORTNOVERS'), and rating ('4.9'). Each item has a 'Распродажа' (Sale) badge. The page also includes promotional banners for 'ПУХОВИК' (Down jacket) and 'ТВОЕ' (TVOE).

Примеры поисковых систем

Desktop Search



Примеры поисковых систем

People Search

The screenshot shows a LinkedIn search results page. At the top, there is a navigation bar with the LinkedIn logo, a dropdown menu labeled 'Люди', the search term 'Константин Воронцов', a magnifying glass icon, a 'Присоединиться' button, and a blue 'Войти' button.

Below the navigation bar, a message indicates '10+ результатов поиска по запросу «Константин Воронцов»' (10+ search results for 'Konstantin Vorontsov').

The search results are listed in a vertical format, each showing a profile picture, the name 'Константин Воронцов', the title, location, and company information:

- Директор по электронной коммерции
Санкт-Петербург
Паромный Центр и еще 2
ЛЭТИ
- Старший менеджер - Фаворит
Россия
Фаворит
- Системный администратор
Россия
Страховая группа "БЛАГОСОСТОЯНИЕ" и еще 2
- Генеральный директор - ОАО УРСАЙТ
Россия
УРСАЙТ
- Руководитель СК Лудус
Санкт-Петербург
Аналитикос
- Начальник отдела земельных отношений
Московская область, Россия
Геокадинжиниринг

ПРИМЕРЫ ПОИСКОВЫХ СИСТЕМ

Question Answer Search

The screenshot shows the Stack Overflow search results page for the query "search engine". The results are displayed in a grid format, each row containing a question, its accepted answer, and some metadata.

- What is the maximum length of a URL in different browsers?**
6074 votes, Accepted. Short answer - de facto limit of 2000 characters if you keep URLs under 2000 characters, they'll work in virtually any combination of client and server software, and any search engine...
http://url/browser
Paul Dixon 301k answered Jan 6, 2009 at 16:22
- How can I prevent SQL injection in PHP?**
9692 votes, Accepted. or a named parameter like :name in the example above) tell the database engine where you want to filter on. ... In the example above, if the \$name variable contains 'Sarah'; DELETE FROM...
PHP, php, mysql, sql, security, sql-injection
Community wiki Theo
- How do search engines deal with AngularJS applications?**
702 votes, 15 answers. I see two issues with AngularJS application regarding search engines and SEO: 1) What happens with custom tags? Do search engines ignore the whole content within those tags? 2) Is there...
209k views
html, angularjs, seo, search-engine, google-search
dariofarzatti 8,648 asked Nov 21, 2012 at 17:44
- Importing files from different folder**
2328 votes. For most programmers coming here from a search engine, this is not the answer you are looking for. ... Typically you would structure your files into packages (see other answers) instead of modi...
python, importerror, python-import
Cameron 98.7k answered Dec 8, 2010 at 2:12
- How does database indexing work?**
4163 votes, Accepted. Whereas with a sorted field, a Binary Search may be used, which has $\log_2 N$ block accesses. ... Now a search using the firstname field can utilize the index to increase performance. This allow...
sql, database, performance, indexing, database-indexes
Xenph Yan 84k answered Aug 4, 2008 at 10:41
- Comparison of full text search engine - Lucene, Sphinx, Postgresql, MySQL?** [closed]
329 votes, 9 answers. I'm building a Django site and I am looking for a search engine... speed ease of use and ease of integration with Django resource requirements - site will be hosted on a VPS, so ideally the sear...
mysql, postgresql, full-text-search, lucene, sphinx
Continuation 13k asked Apr 10, 2009 at 10:38

Hot Network Questions

- Cauchy's theorem : Homotopy vs Homology
- Why is chmod 777 so bad if I'm the only user on the system?
- Why does the 'target' attribute in HTML require a string starting with an underscore, such as "_blank"? Why not simply "blank"?
- Gradient eye color
- How to show/hide geometry node panel group or inputs in modify tab?
- How to step through different diodes
- What is the crest on this 'book' cover?
- Invitation letter Problem in Sweden
- How to measure generality if there is a transcendental optimizer?
- How often do authors rewrite chapters?
- Covering with small measure a subsequence of a sequence of small-measure sets
- 1950s/1960s story about a comet coming to hit the earth and it brings world peace, then astronomer realizes it will miss earth
- Jewish pirate buried next to ARIZAL
- Is it France, Netherlands or Poland?
- Can a business require a document be notarised by a specific notary?
- Does Combining Predictors Always Improve AUC?
- Did the Biden administration earmark \$50M for condoms in Gaza?
- Who is the lady at the end of 12 Monkeys?

Структурированная и неструктурированная информация

Structured (Databases) vs. Unstructured (IR)

- Чаще всего информация хранится в табачном виде

Country	City	Area, sq.km.
Россия	Москва	2511
Россия	Казань	516
Франция	Париж	105

- Допускается поиск по полному совпадению (для текстовых полей) и по промежуткам (для числовых полей)
Например, «Country == ‘Россия’ AND Area > 2000»

Structured (Databases) vs. Unstructured (IR)

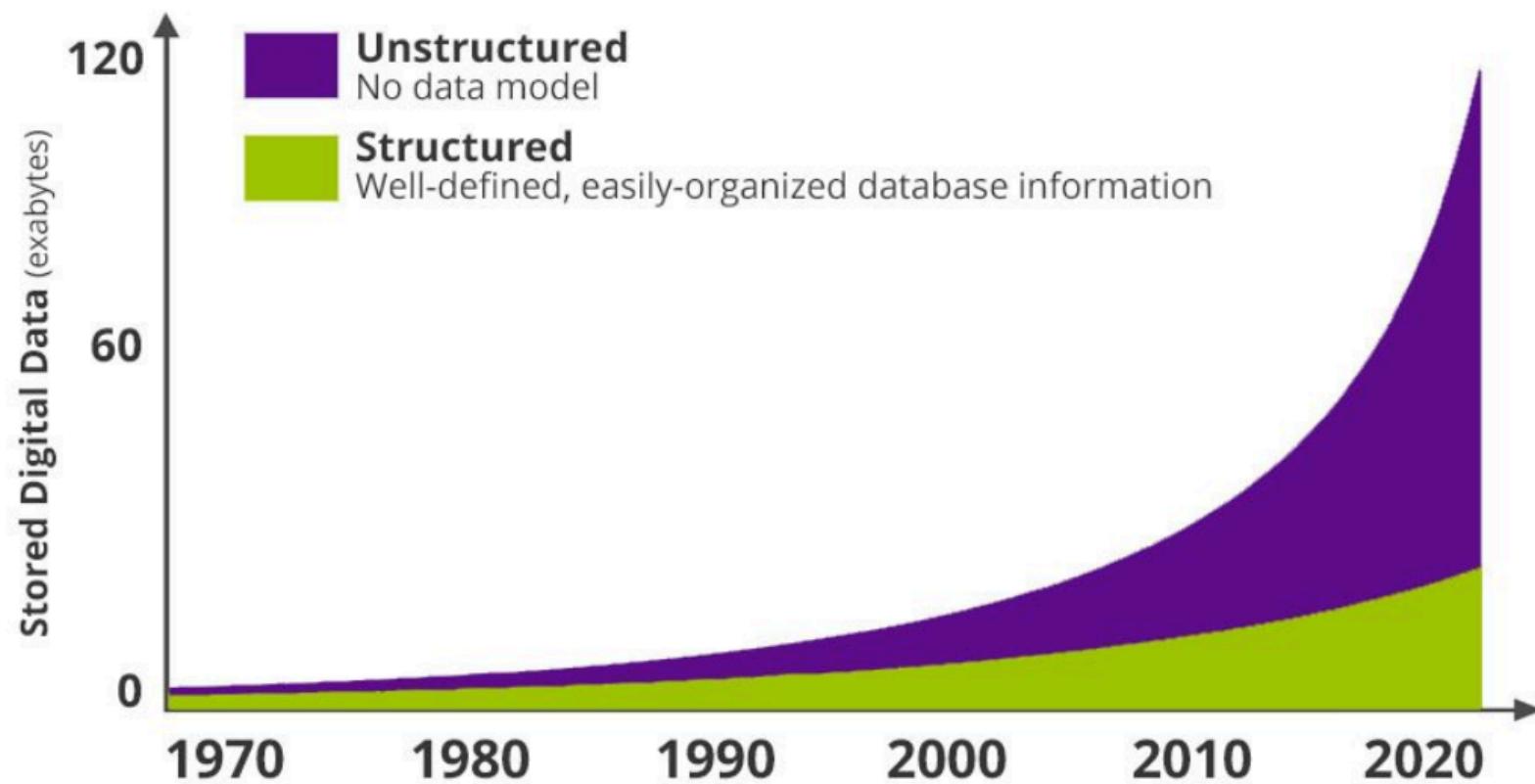
- Чаще всего информация хранится в текстовом виде

«Преступление и наказание» (рус. дореф. Преступленіе и наказаніе) — социально-психологический и социально-философский роман Фёдора Михайловича Достоевского, над которым писатель работал в 1865—1866 годах
- Допускается поиск по вхождению строки в текст страницы
Например, найти все страницы, где встречается «Преступление и наказание»

Structured (Databases) vs. Unstructured (IR)

	IR	Database
Query	Natural (free formed text)	Artificial (sql)
Query specification	Incomplete	Complete
Matching	Vague	Exact
Items wanted	Relevant	All matched
Ranking	Based on relevance	Not applicable

Structured (Databases) vs. Unstructured (IR)



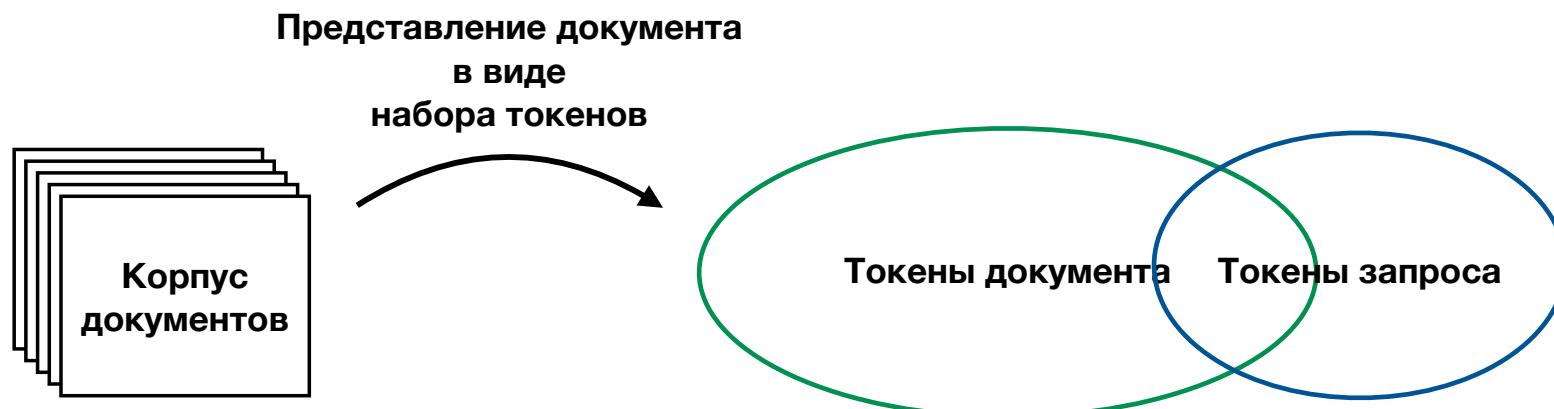
Вызовы IR

- Люди не всегда знают, что они хотят
- Люди не всегда знают, как сформулировать то, что они хотят
- Компьютер не понимает естественный язык
- Поисковая система может только догадываться о том, что релевантно
- Поисковая система может только догадываться об удовлетворенности пользователя

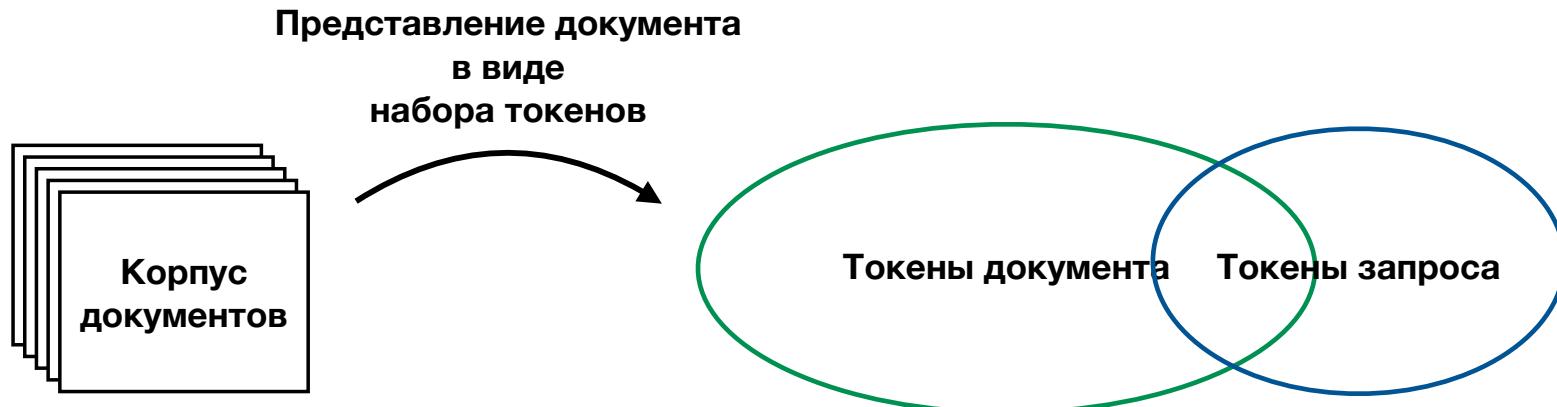
Bag-of-words представление

Bag-of-words представление

Задача: по данному набору слов (токенов, термов) запроса найти документы, содержащие эти токены.



Bag-of-words представление



BOW-поиск:

- Документ представляется набором токенов
- Запрос представляется набором токенов
- Находится пересечение между этими наборами токенов

Представление документа в BOW

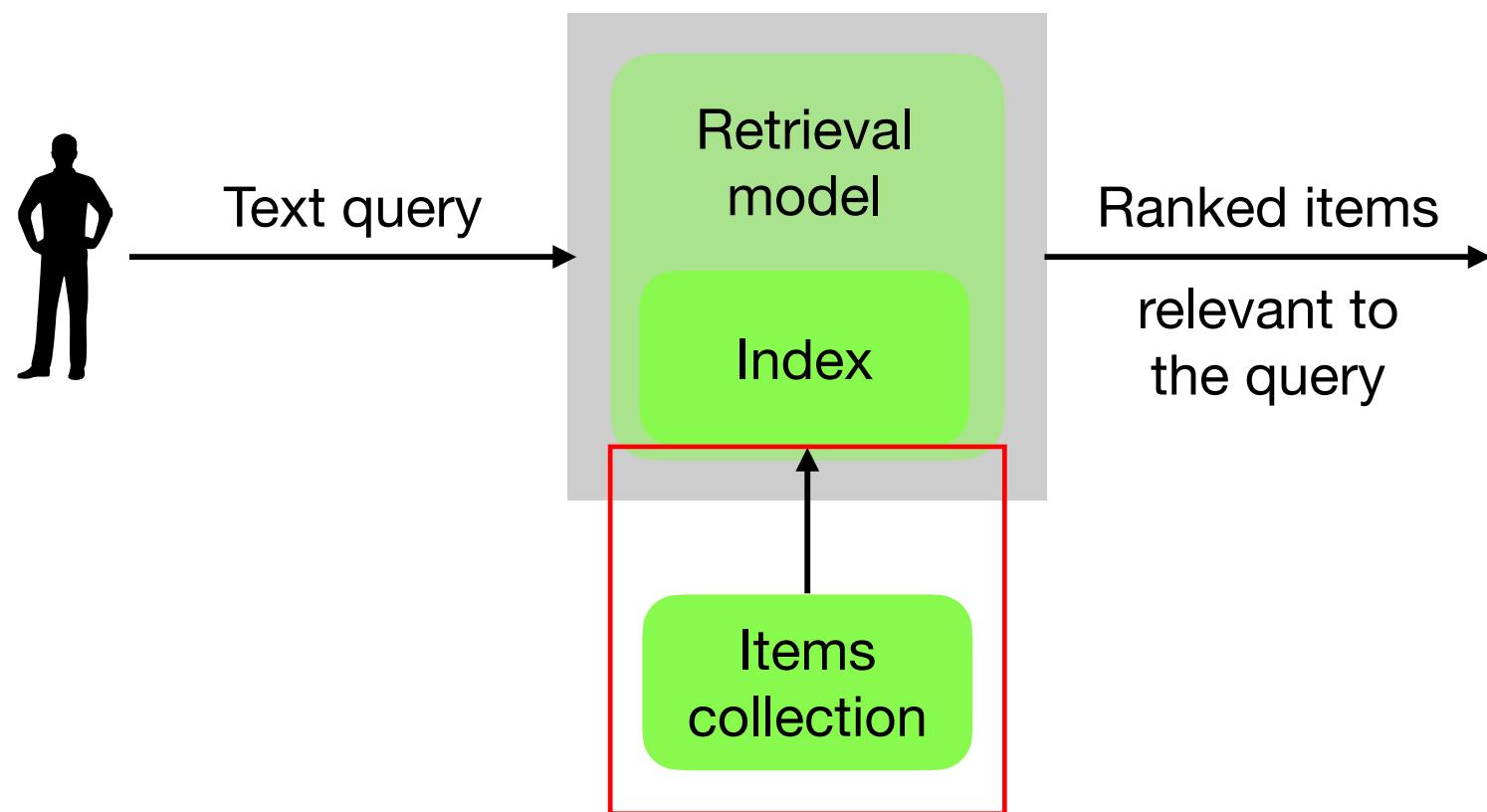
Задача состоит в том, чтобы хранить набор документов (корпус) таким образом, чтобы по набору токенов (запросу) мы могли быстро найти те документы, которые содержат токены запроса.

Стадии:

- Обработка текста;
- Размещение документов для быстрого доступа — индексация.

Обработка текста

Поисковая система



Обработка текста

Мастер и Маргарита

Материал из Википедии — свободной энциклопедии

Стабильная версия, проверенная 4 января 2025.

У этого термина существуют и другие значения, см. [Мастер и Маргарита \(значения\)](#).

«Мáстер и Маргáрita» — роман Михаила Афанасьевича Булгакова, работа над которым началась, по одним данным, в 1928 году, по другим — в 1929-м и продолжалась вплоть до смерти писателя в марте 1940 года. Роман относится к незавершённым произведениям; редактирование и сведение воедино черновых записей осуществляла после смерти мужа вдова писателя — Елена Сергеевна. Первоначальная версия произведения, задуманного как «роман о дьяволе»^[1], имела рабочее название «Копыто инженера»^{[2][3][комм. 1]} и была уничтожена Булгаковым в 1930 году. В последующих редакциях среди героев произведения появились автор романа о Понтии Пилате и его возлюбленная. Окончательное название — «Мастер и Маргарита» — закрепилось в 1937 году.

Первая публикация романа в сокращённом (цензурированном) виде была осуществлена в 1966—1967 годах (журнал «Москва», предисловие Константина Симонова, послесловие Абрама Вулиса). Первое однтомное издание книги на русском языке с тем же цензурированным текстом, опубликованным в журнале «Москва», вышло в 1967 году (издательство YMCA-Press, Париж)^[4]. В 1969 году издательство «Посев» (Франкфурт-на-Майне) выпустило книгу, соединив публикацию в журнале «Москва» и текст «Купюр из романа „Мастер и Маргарита“ (1929—1940)», подготовленных Е. С. Булгаковой, выделив при этом цензурные изъятия курсивом. В СССР книжный вариант без купюр (с редакторской правкой А. А. Саакянц) увидел свет в 1973 году (издательство «Художественная литература»). В 2014 году был опубликован основной текст романа в составе полного собрания черновиков, представляющего собой текстологическую монографию, основанную на более чем 10-летнем изучении архивных материалов.

В «Мастере и Маргарите» получили продолжение и завершились важнейшие для Булгакова мотивы и стилистические искания: ироническое **остранение** обыденного, сочетание **фантастики** и **обыденности**, **сатирический гротеск**, конфликт творца и эпохи ►. «Мастер и Маргарита» сочетает в себе сатиру и философскую условность, **фэнтези**^[5], **библейские** образы, мифологические **архетипы**^[6], элементы **детектива**, авантюрного романа и др ►. Эти стилистическая синтетичность и многопроблемность — причины того, что жанр «Мастера и Маргариты» определить трудно: роман-притча, роман-утопия, сатирический роман, исторический, приключенческий, философский — каждое из этих определений верно, но не полно ►.

Обработка текста

- Приведение к нижнему регистру: «Роман» - «роман»
- Удаление лишних символов: «БулгаковыM,» - «БулгаковыM»
- Удаление лишних слов: «роман о дьяволе» - «роман дьяволе»
- Токенизация: «Михаила Афанасьевича Булгакова» - [«Михаила», «Афанасьевича», «Булгакова»]
- Лингвистическая обработка: «году» - «ГОД»

Приведение к нижнему регистру

Before	Окончательное название — «Мастер и Маргарита» — закрепилось в 1937 году.
After	окончательное название — «мастер и маргарита» — закрепилось в 1937 году.

Удаление лишних символов

Before	окончательное название — «мастер и маргарита» — закрепилось в 1937 году.
After	окончательное название мастер и маргарита закрепилось в 1937 году

Удаление лишних слов

Before	окончательное название мастер и маргарита закрепилось в 1937 году
After	окончательное название мастер маргарита закрепилось 1937 году

Токенизация

- Разделение по пробелу
- Разделение по специальным символам (пробелы, знаки препинания и т.д.)
- ВРЕ-токенизация

Токенизация

Before	окончательное название мастер маргарита закрепилось 1937 году
After	[окончательное, название, мастер, маргарита, закрепилось, 1937, году]

Лингвистическая обработка

Проблема:

- Документ: Автором романа Мастер и Маргарита является Булгаков.
- Запрос: Роман Булгакова

Лингвистическая обработка

Проблема:

- Документ: Автором романа Мастер и Маргарита является Булгаков.
- Запрос: Роман Булгакова

2 основных подхода к нормализации текста:

- Стемминг
- Лемматизация

Стемминг

- Процесс нахождения основы слова (**стем**) для заданного исходного слова
- Примеры:
 - машину, машины, машинный — машин
 - романы, роману, романний — роман
- 2 типа:
 - основанный на правилах: «-ий» - «»; «ЮТ» - «»; ...
 - статистический: «institute», «institution» - «institut»

Лемматизация

- Процесс приведения словоформы к **лемме** – её нормальной (словарной) форме
- Примеры:
 - романа, романов, романы – роман
 - закрепилось, закрепился – закрепиться
- Необходимы знания лингвистики – дорого

Лингвистическая обработка - стемминг

Before	окончательное название мастер маргарита закрепилось 1937 году
After	[окончательн, назван, мастер, маргарит, закреп, 1937, год]

Лингвистическая обработка - лемматизация

Before	окончательное название мастер маргарита закрепилось 1937 году
After	[окончательный, название, мастер, маргарита, закрепиться, 1937, год]

Представление документа

Простейший поиск

Поиск подстроки в документах корпуса

- Поиск за линейное время - нужно обойти каждый документ целиком
- А если таких документов не 100, а 100 тысяч?

Простейший поиск

Поиск подстроки в документах корпуса

- Поиск за линейное время - нужно обойти каждый документ целиком
- А если таких документов не 100, а 100 тысяч?
- Нужен более эффективный механизм!

Простейший поиск

Поиск подстроки в документах корпуса

- Поиск за линейное время - нужно обойти каждый документ целиком
- А если таких документов не 100, а 100 тысяч?
- Нужен более эффективный механизм!
- **Решение** – индексация документов – для каждого документа d будем хранить термы t , которые встретились в этом документе.

Incidence matrix

- Матрица $M \times N$, где:
 - M – количество уникальных токенов в словаре;
 - N – количество документов в корпусе;

	мама мыла раму	мама мыла пол	деревянная рама
мама	1	1	0
мыть	1	1	0
рама	1	0	1
пол	0	1	0
деревянный	0	0	1

Incidence matrix

- Вектор представления токена (term incidence vector, TIV) – вектор-индикатор встречаемости данного токена в документах корпуса;
- Вектор представления документа (document incidence vector, DIV) – вектор-индикатор встречаемости токенов словаря в данном документе;

	мама мыла раму	мама мыла пол	деревянная рама
мама	1	1	0
мыть	1	1	0
рама	1	0	1
пол	0	1	0
деревянный	0	0	1

Обработка запроса – Boolean Retrieval

- Query = TIV(мама) AND TIV(мыть)
- Result = [«мама мыла раму», «мама мыла пол»]

	мама мыла раму	мама мыла пол	деревянная рама
мама	1	1	0
мыть	1	1	0
рама	1	0	1
пол	0	1	0
деревянный	0	0	1

Обработка запроса – Boolean Retrieval

- Query = TIV(мама) AND TIV(мыть)
- Result = [«мама мыла раму», «мама мыла пол»]
- Операторы: AND, OR, NOT

	мама мыла раму	мама мыла пол	деревянная рама
мама	1	1	0
мыть	1	1	0
рама	1	0	1
пол	0	1	0
деревянный	0	0	1

Boolean retrieval

Преимущества:

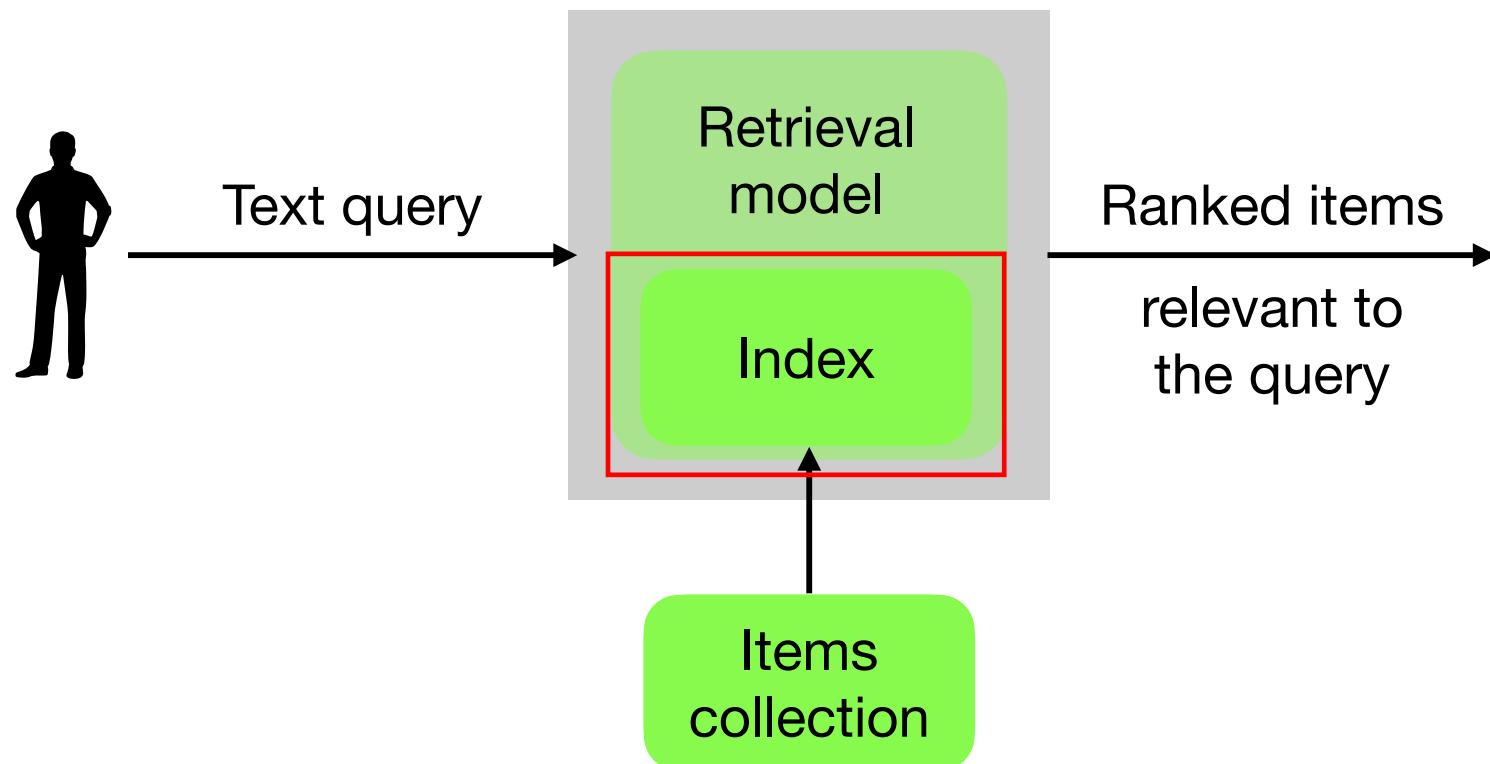
- Легок для обработки системой
- Прозрачен и контролируем пользователем
- Контроль количества найденных документов через баланс AND и OR

Недостатки:

- Все токены одинаково важны
- Сложно сформулировать запрос
- Большая размеренность матрицы

Обратный индекс

Поисковая система



Обратный индекс

- Для каждого токена храним вектор встречаемости данного токена в документах корпуса

	мама мыла раму	мама мыла пол	деревянная рама
мама	1	1	0
мыть	1	1	0
рама	1	0	1
пол	0	1	0
деревянный	0	0	1

Обратный индекс

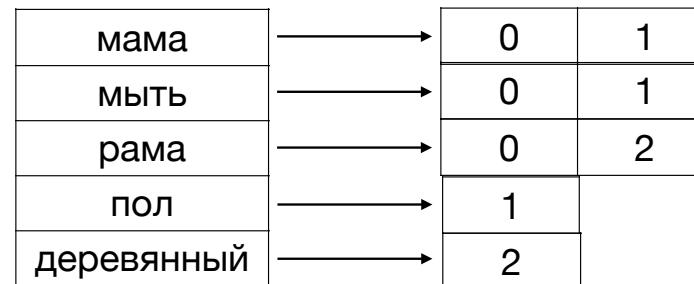
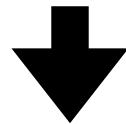
- Для каждого токена храним вектор встречаемости данного токена в документах корпуса

	мама мыла раму	мама мыла пол	деревянная рама
мама	1	1	0
мыть	1	1	0
рама	1	0	1
пол	0	1	0
деревянный	0	0	1

- А что если хранить для каждого токена хранить только индексы документов, в которых этот токен встречается?

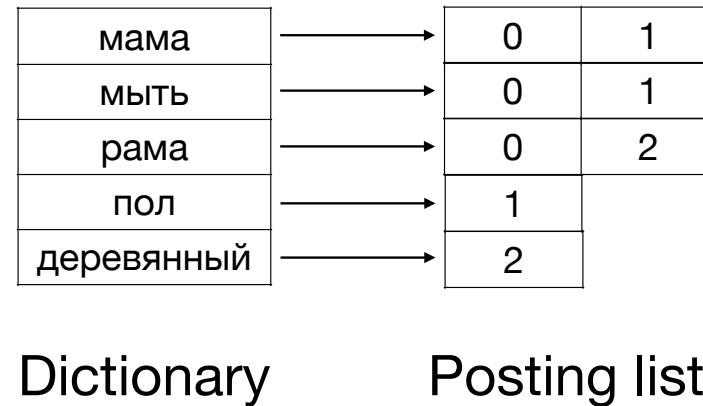
Обратный индекс

	мама мыла раму	мама мыла пол	деревянная рама
мама	1	1	0
мыть	1	1	0
рама	1	0	1
пол	0	1	0
деревянный	0	0	1



Обратный индекс

Терминология



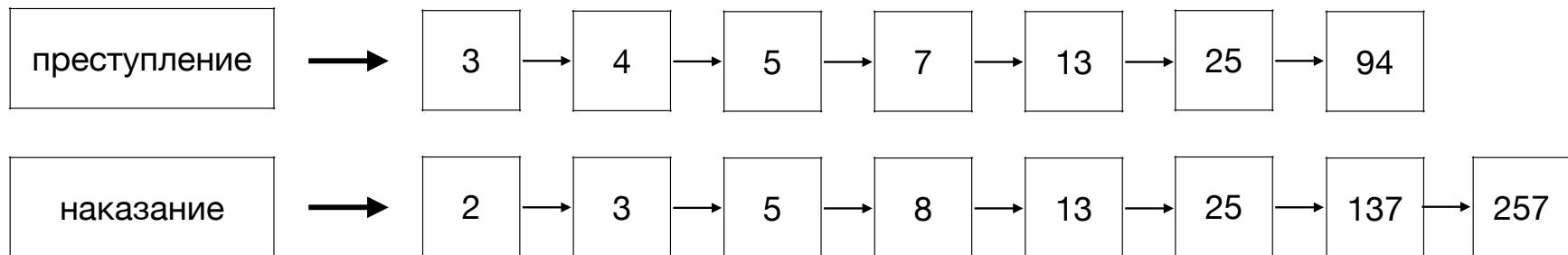
- Прямой индекс: документ со списком токенов, встречающихся в этом документе: $\text{document} \rightarrow \text{terms}$
- Обратный индекс: токен со списком документов, в которых он встречается: $\text{term} \rightarrow \text{documents}$

Обработка запроса – Inverted Index

Обработка запроса – Inverted Index

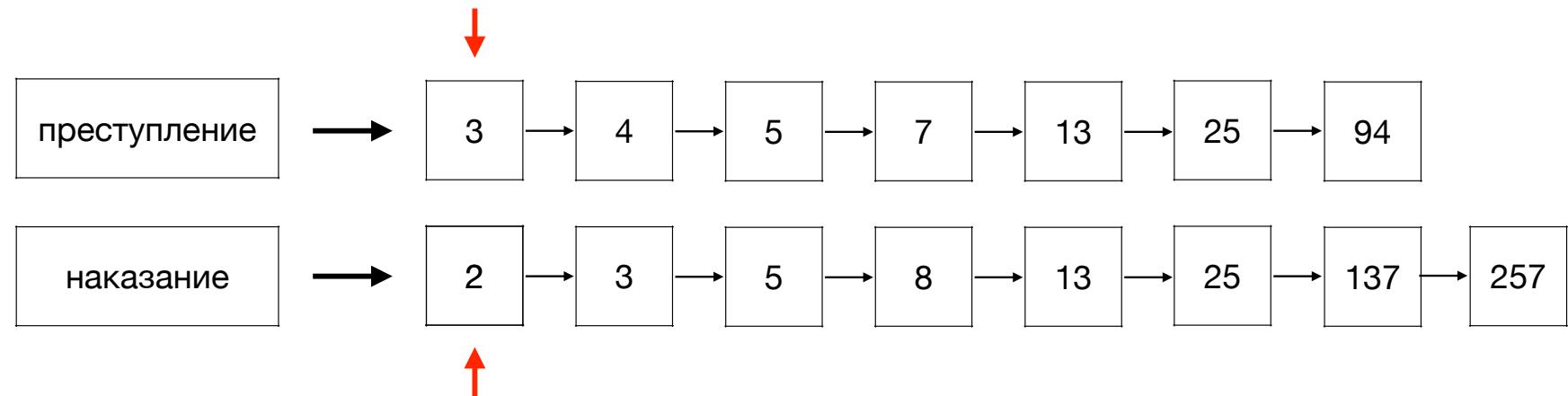
Преступление AND Наказание

- Получаем posting list для «преступление»
- Получаем posting list для «наказание»



Обработка запроса – Inverted Index

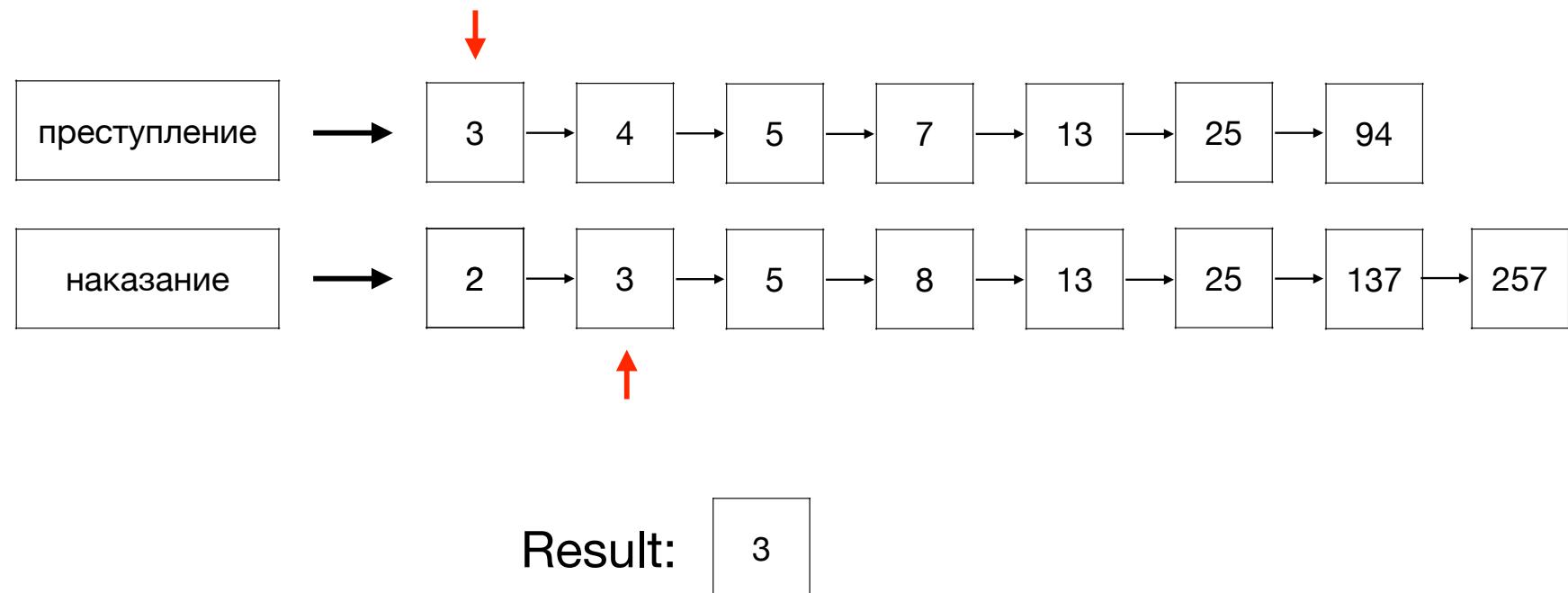
Слияние списков



Result:

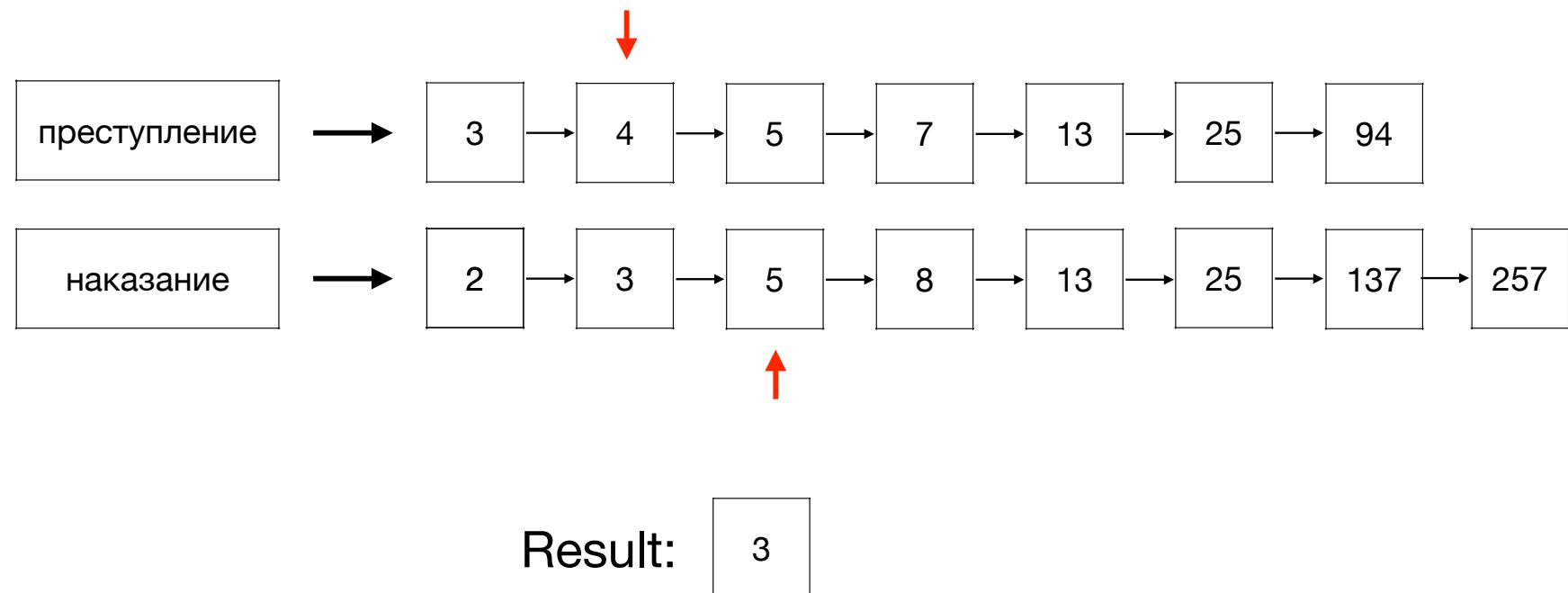
Обработка запроса – Inverted Index

Слияние списков



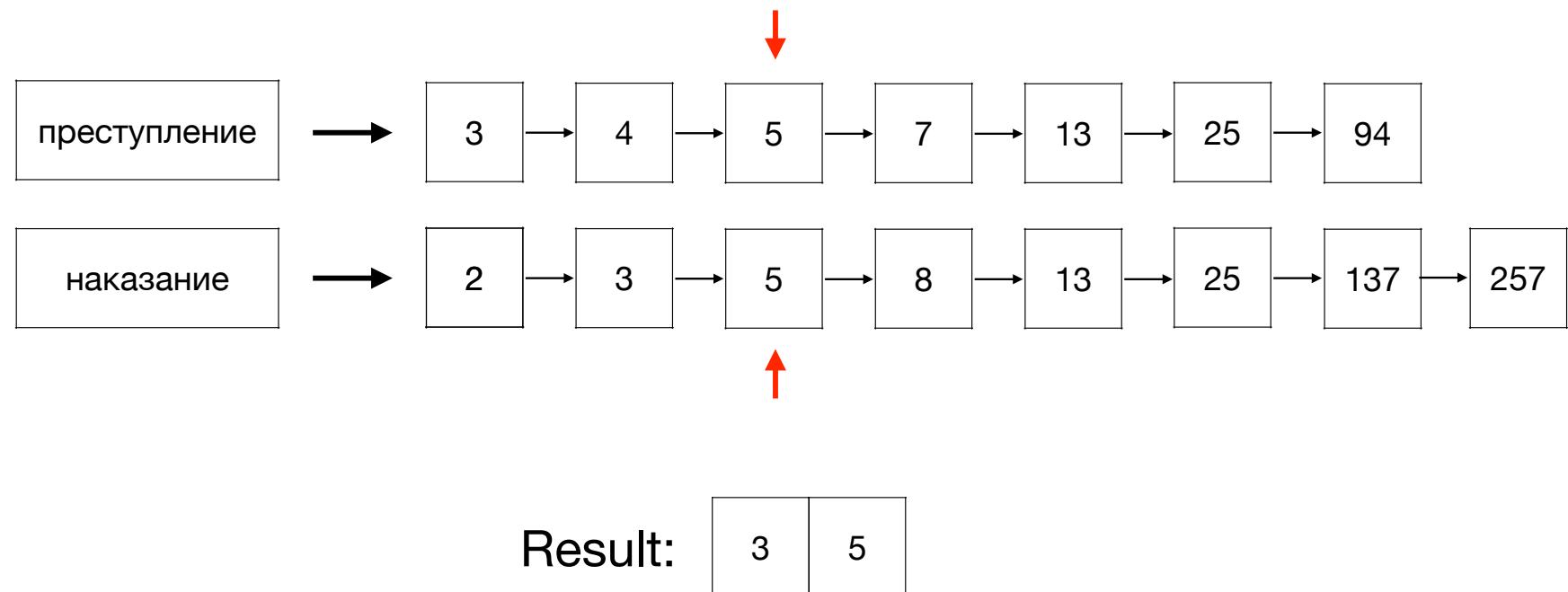
Обработка запроса – Inverted Index

Слияние списков



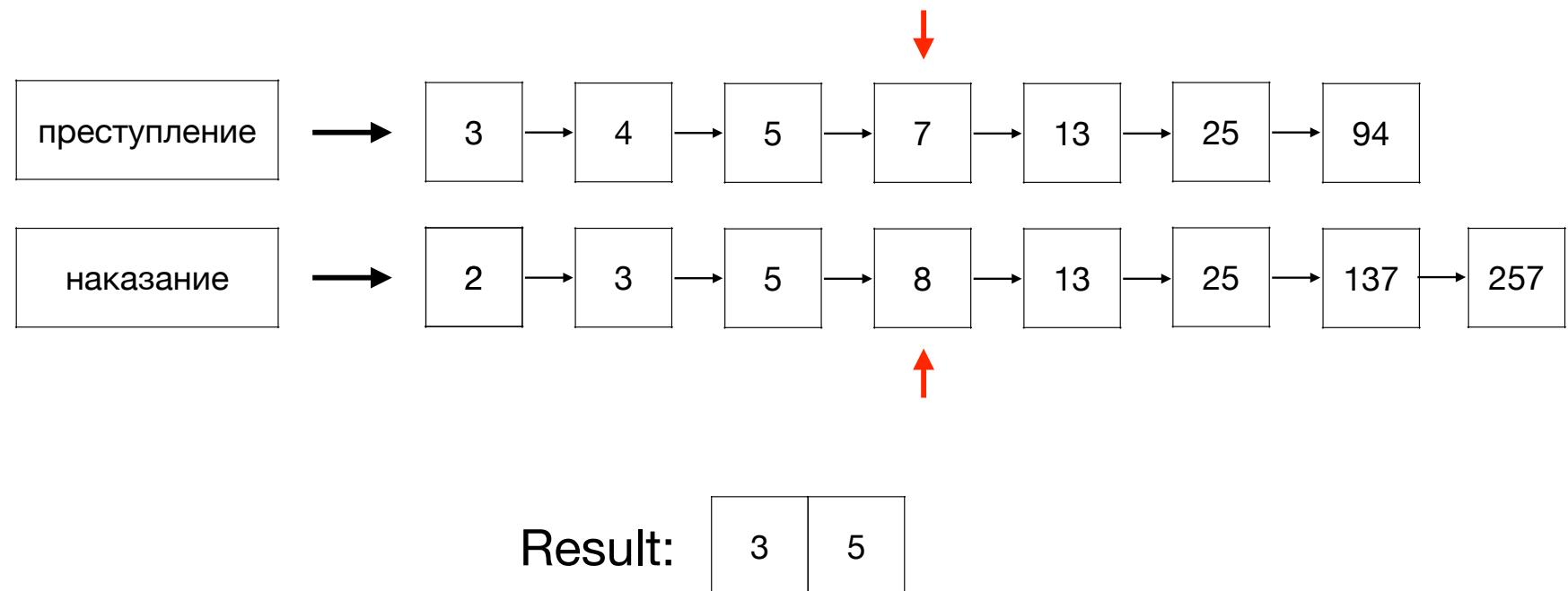
Обработка запроса – Inverted Index

Слияние списков



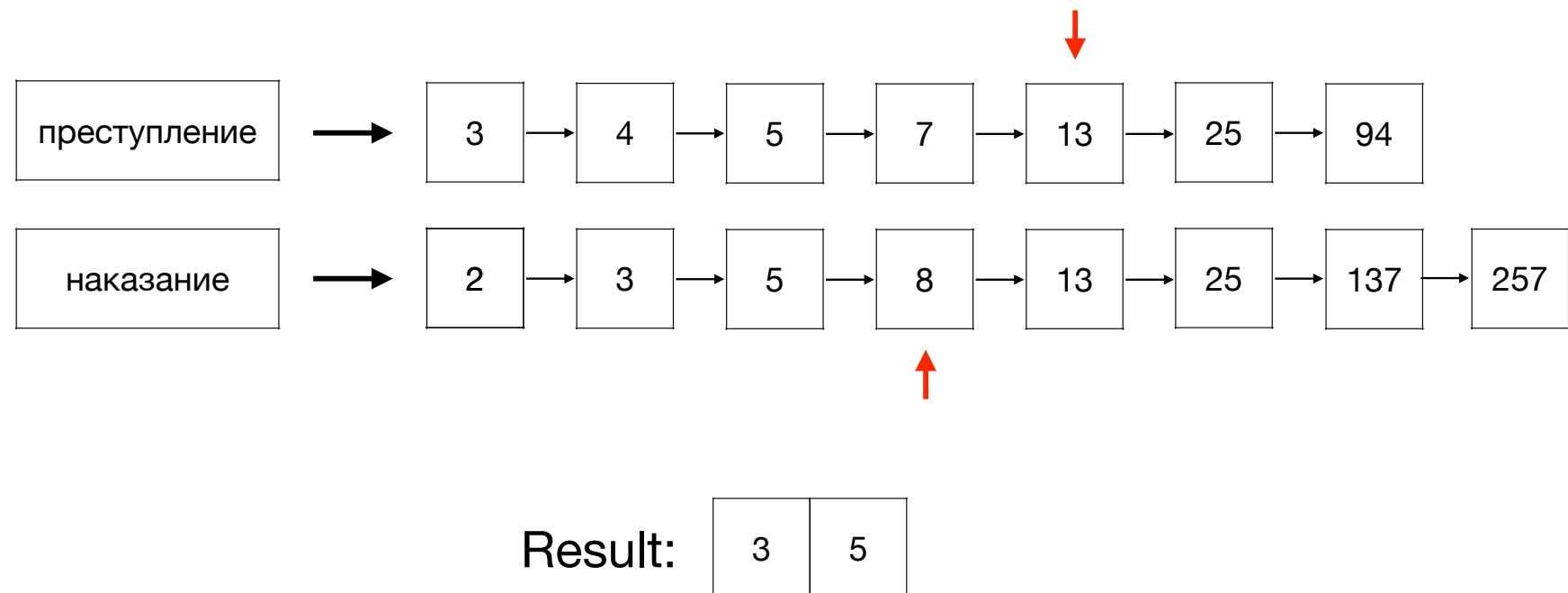
Обработка запроса – Inverted Index

Слияние списков



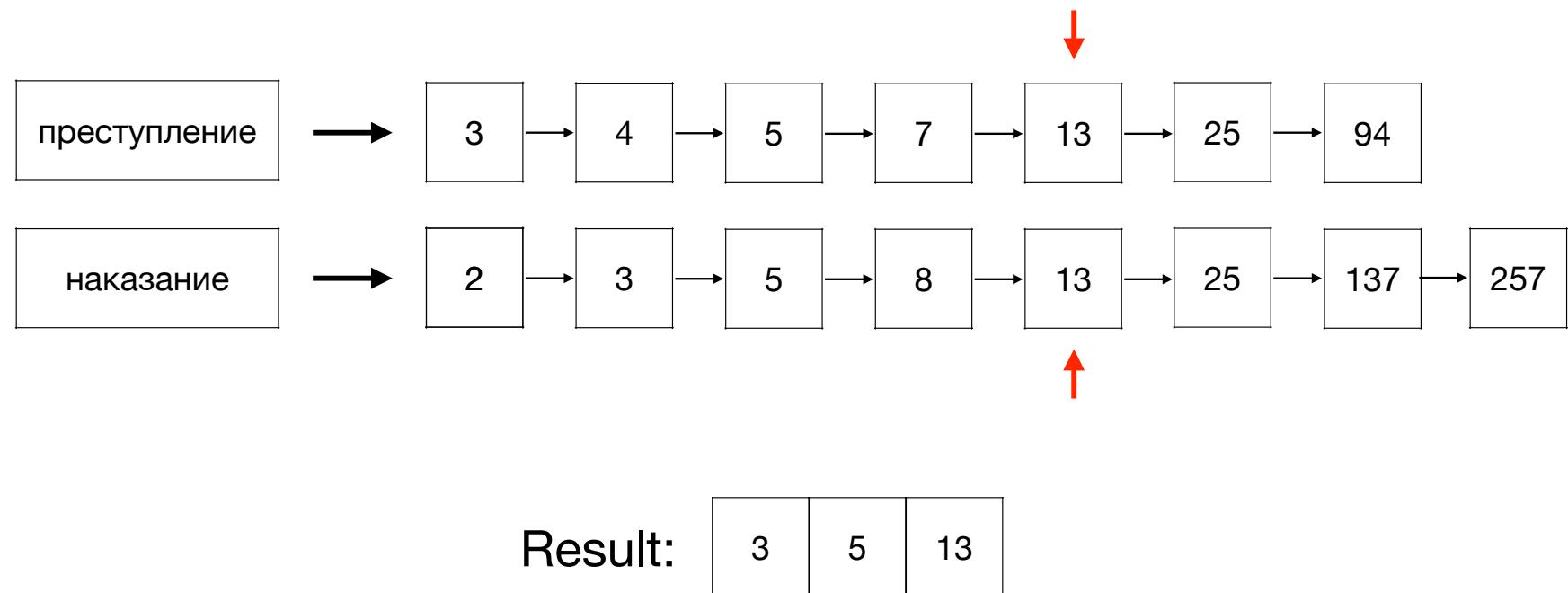
Обработка запроса – Inverted Index

Слияние списков



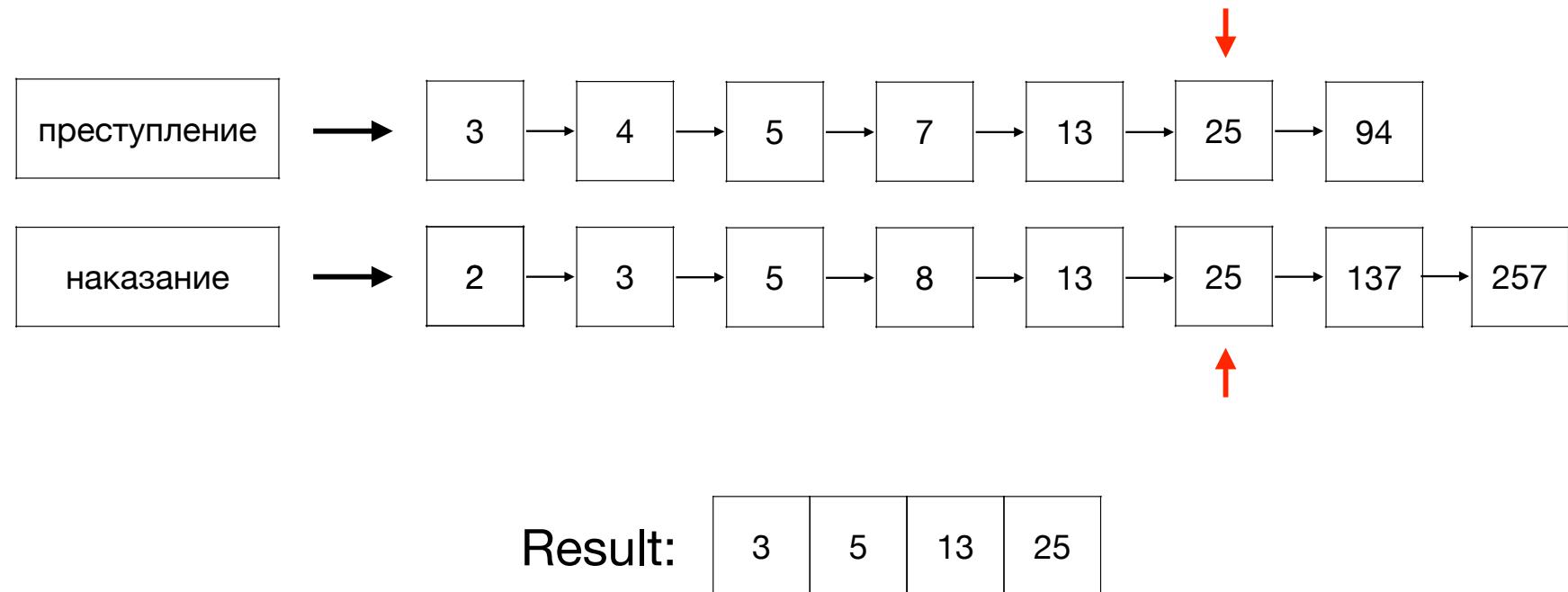
Обработка запроса – Inverted Index

Слияние списков



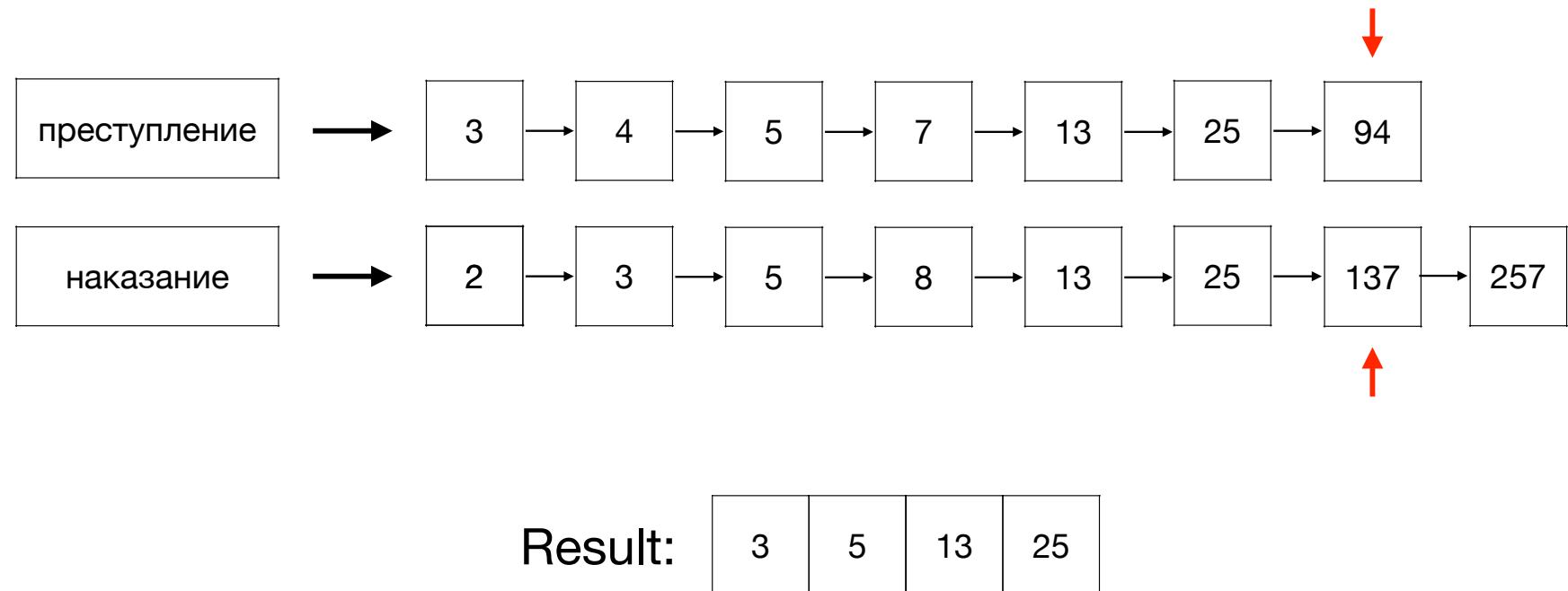
Обработка запроса – Inverted Index

Слияние списков



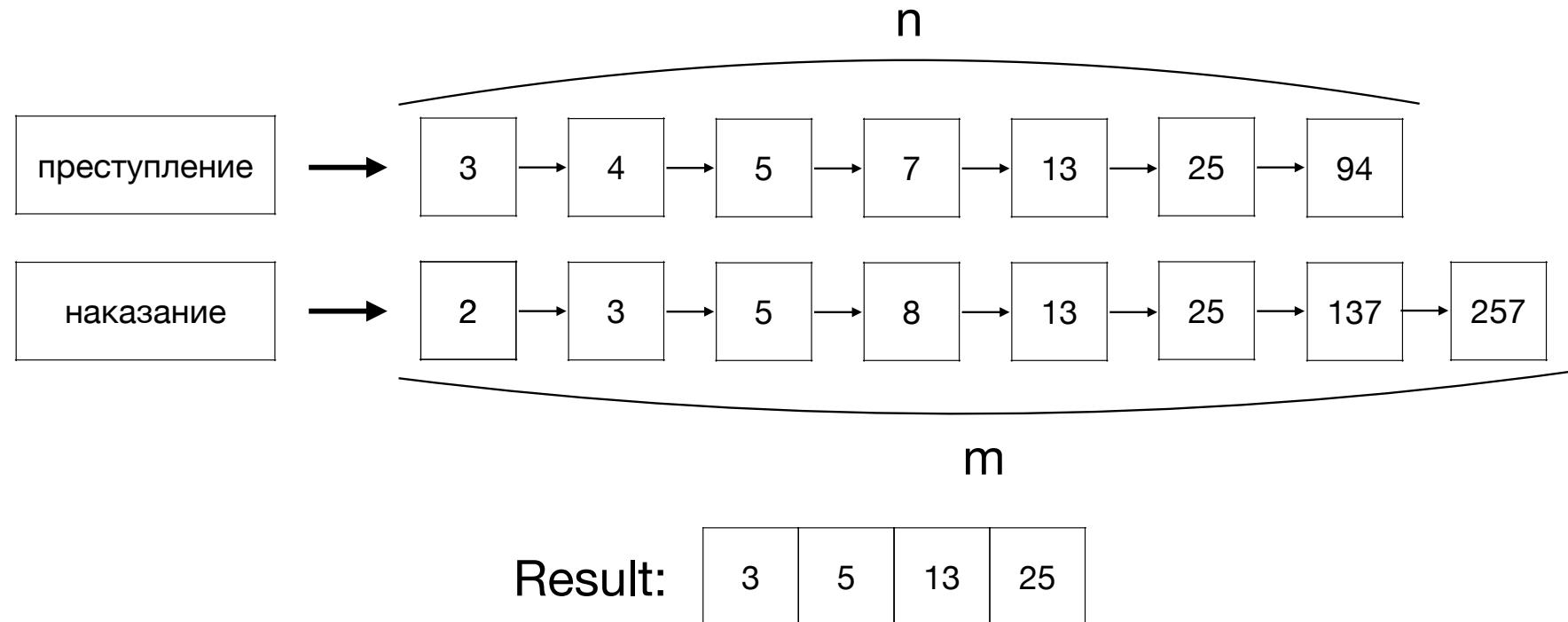
Обработка запроса – Inverted Index

Слияние списков



Обработка запроса – Inverted Index

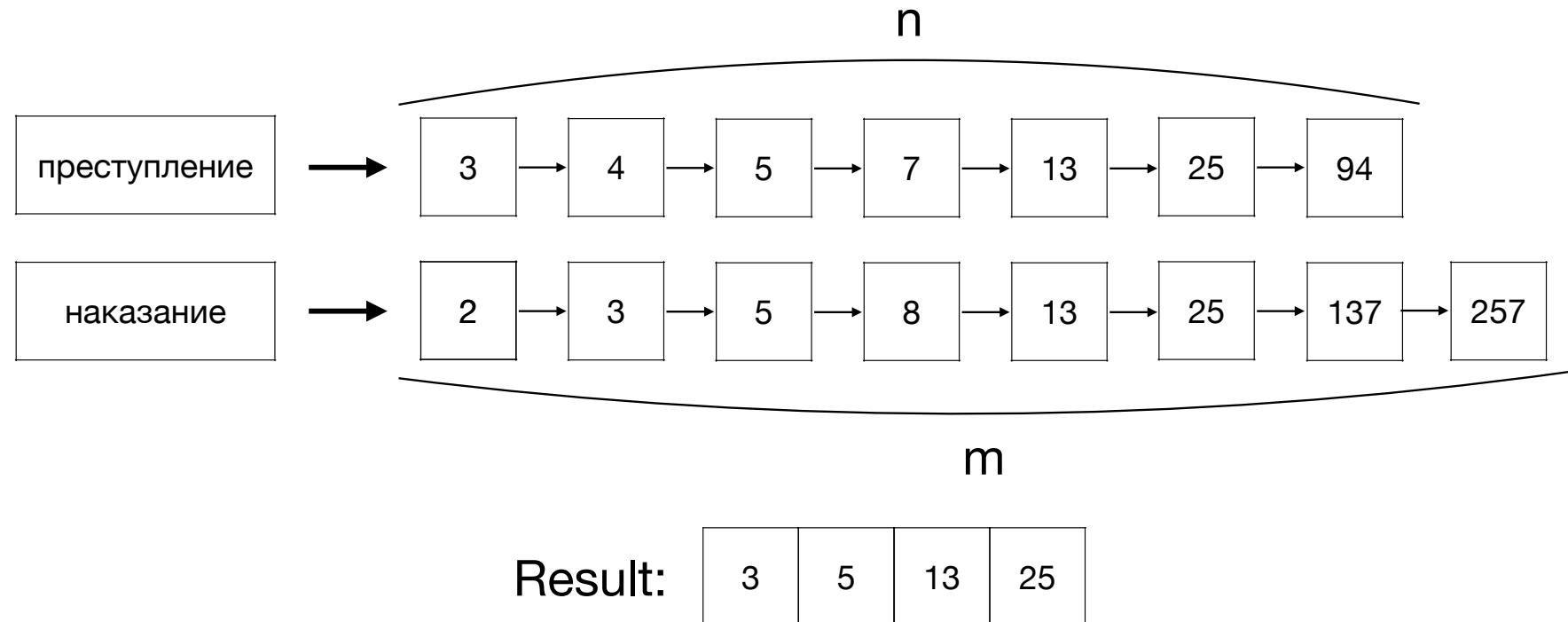
Слияние списков



Сложность по времени: $O(n+m)$

Обработка запроса – Inverted Index

Слияние списков

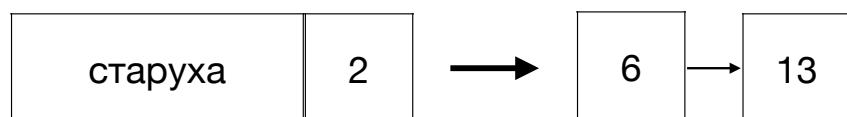
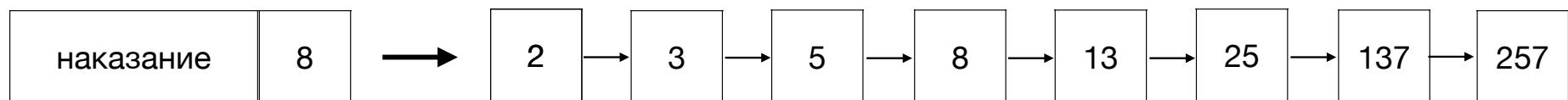
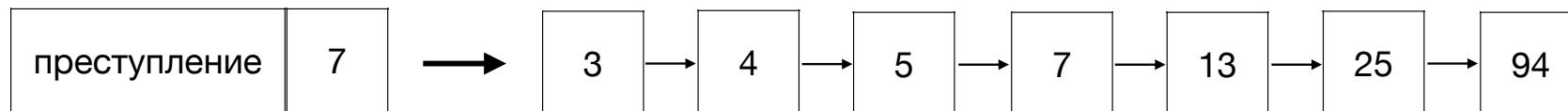


Сложность по времени: $O(n+m)$

Важно: списки отсортированы по doc_id, иначе сложность $O(\max(x, y)^2)$

Обработка запроса – Inverted Index

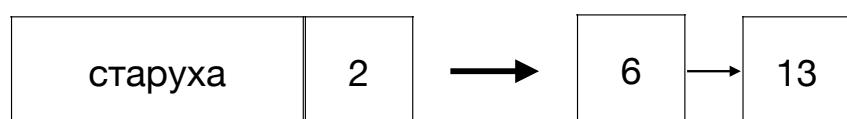
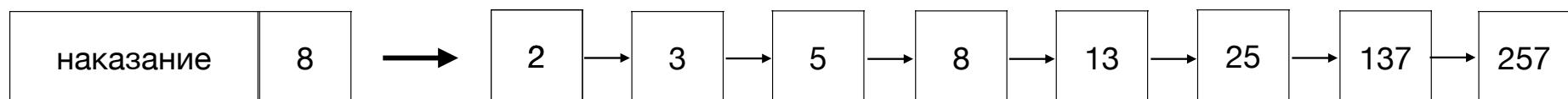
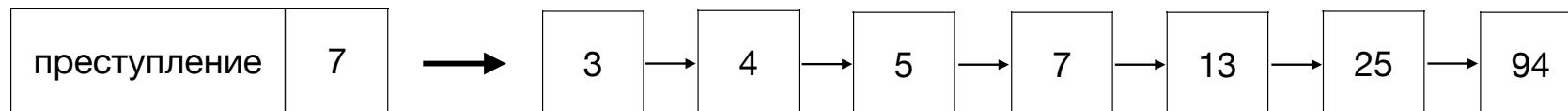
Слияние списков – оптимизация – document frequency (df)



Преступление AND Наказание AND Старуха

Обработка запроса – Inverted Index

Слияние списков – оптимизация – document frequency (df)

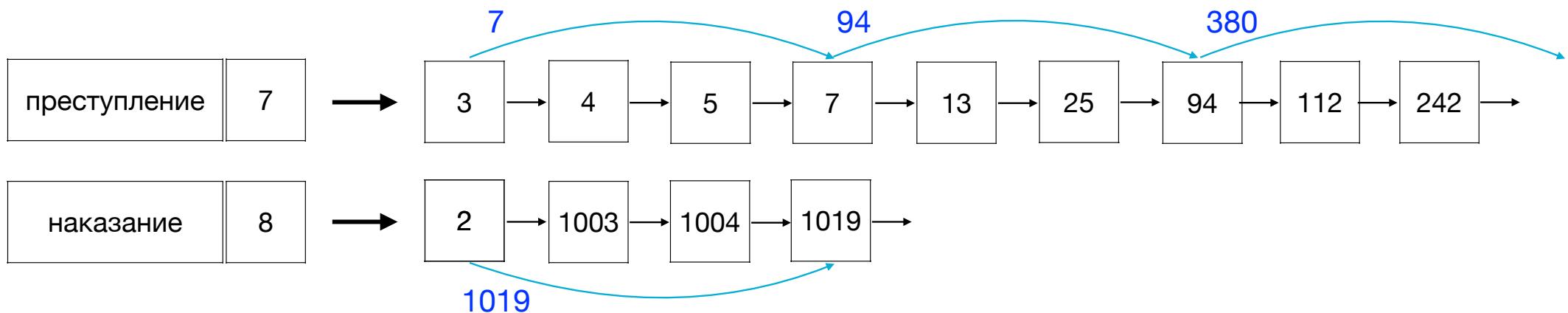


Преступление AND Наказание AND Старуха

Преступление AND (Наказание AND Старуха)

Обработка запроса – Inverted Index

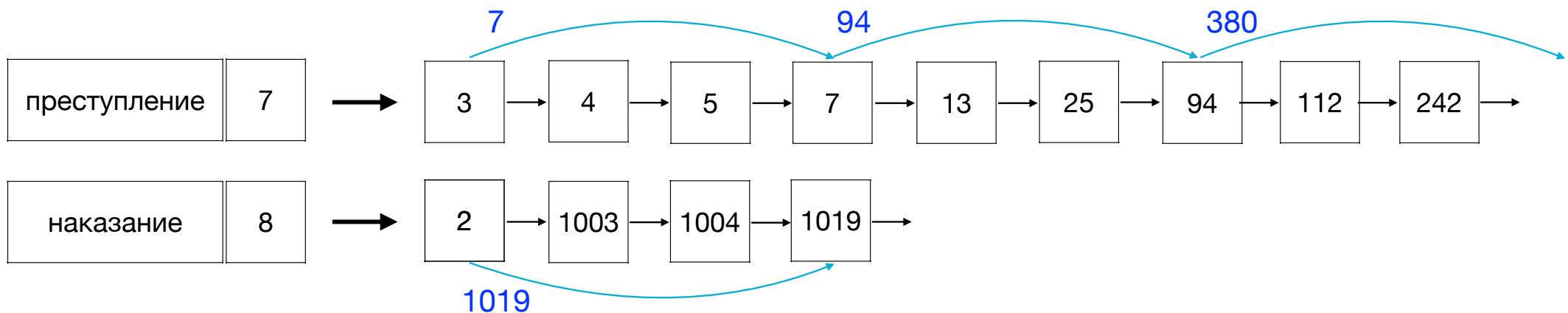
Слияние списков – оптимизация – skip pointers



- Дают возможность не обрабатывать часть списка

Обработка запроса – Inverted Index

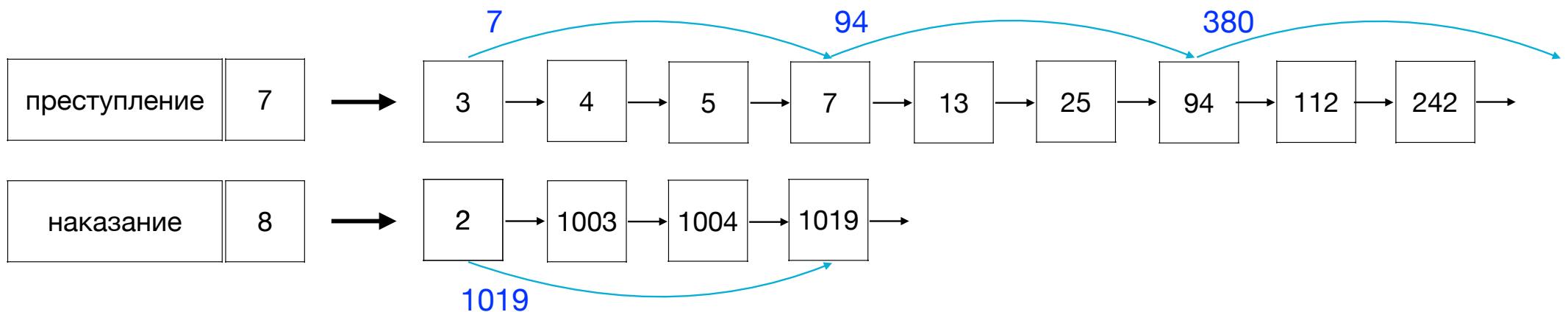
Слияние списков – оптимизация – skip pointers



- Дают возможность не обрабатывать часть списка
- Как часто расставлять указатели?

Обработка запроса – Inverted Index

Слияние списков – оптимизация – skip pointers



- Дают возможность не обрабатывать часть списка
- Как часто расставлять указатели?
- Для списка длины N рекомендуется расставлять их на каждые \sqrt{N} позиций — хорошо работает на практике

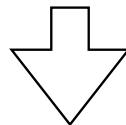
Индексация

Индексация

doc_1	doc_2
Преступление и наказание - роман Достоевского.	Раскольников совершил преступление и понес наказание

Индексация

doc_1	doc_2
Преступление и наказание - роман Достоевского.	Раскольников совершил преступление и понес наказание



Лингвистическая обработка

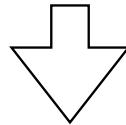
doc_1	doc_2
[преступление, наказание, роман, достоевский]	[раскольников, совершить, преступление, понести, наказание]

Индексация

doc_1	doc_2
[преступление, наказание, роман, достоевский]	[раскольников, совершить, преступление, понести, наказание]

Индексация

doc_1	doc_2
[преступление, наказание, роман, достоевский]	[раскольников, совершить, преступление, понести, наказание]



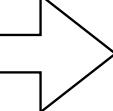
token	doc_id
преступление	1
наказание	1
роман	1
достоевский	1
раскольников	2
совершить	2
преступление	2
понести	2
наказание	2

Индексация

token	doc_id
преступление	1
наказание	1
роман	1
достоевский	1
раскольников	2
совершить	2
преступление	2
понести	2
наказание	2

Индексация

token	doc_id
преступление	1
наказание	1
роман	1
достоевский	1
раскольников	2
совершить	2
преступление	2
понести	2
наказание	2

order by
token,
doc_id


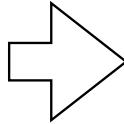
token	doc_id
достоевский	1
наказание	1
наказание	2
понести	2
преступление	1
преступление	2
раскольников	2
роман	1
совершить	2

Индексация

token	doc_id
достоевский	1
наказание	1
наказание	2
понести	2
преступление	1
преступление	2
раскольников	2
роман	1
совершить	2

Индексация

token	doc_id
достоевский	1
наказание	1
наказание	2
понести	2
преступление	1
преступление	2
раскольников	2
роман	1
совершить	2



Индексация

- Лингвистическая обработка
 - Приведение к нижнему регистру
 - Удаление лишних символов
 - Удаление лишних слов
 - Токенизация
 - Лемматизация / стемминг
- Сортировка пар (токен, документ)
- Группировка по токенам, подсчет document frequency, составление posting lists

Жизненный цикл обратного индекса

- **Построение индекса:** для каждого токена корпуса собирается свой массив документов - *posting list* – **indexing time**
- **Обработка запроса:** построенный на первом этапе индекс используется для обработки поисковых запросов – **searching time**

Жизненный цикл обратного индекса

- **Построение индекса:** для каждого токена корпуса собирается свой массив документов - *posting list* – **indexing time**
- **Обработка запроса:** построенный на первом этапе индекс используется для обработки поисковых запросов – **searching time**
- **Цель:** как можно быстрее обрабатывать поисковые запросы

Жизненный цикл обратного индекса

- **Построение индекса:** для каждого токена корпуса собирается свой массив документов - *posting list* – **indexing time**
- **Обработка запроса:** построенный на первом этапе индекс используется для обработки поисковых запросов – **searching time**
- **Цель:** как можно быстрее обрабатывать поисковые запросы
 - Быстро находить токен в словаре
 - Быстро выполнять слияние листов документов

Как хранится словарь

- Что нужно хранить:
 - Токен
 - Document frequency
 - Ссылку на posting list
- Два подхода к хранению словаря
 - Sort-based
 - Hash-based
- Выбор зависит от:
 - Количество токенов коллекции
 - Static / dynamic index



Sort-based dictionary

- Indexing time — храним токены в виде отсортированного массива или бинарного дерева поиска
- Searching time — бинарный поиск по массиву или обход дерева

Sort-based dictionary

- Indexing time — храним токены в виде отсортированного массива или бинарного дерева поиска
- Searching time — бинарный поиск по массиву или обход дерева
- Алгоритмическая сложность поиска: $O(\log(\text{size}(\text{dictionary})))$

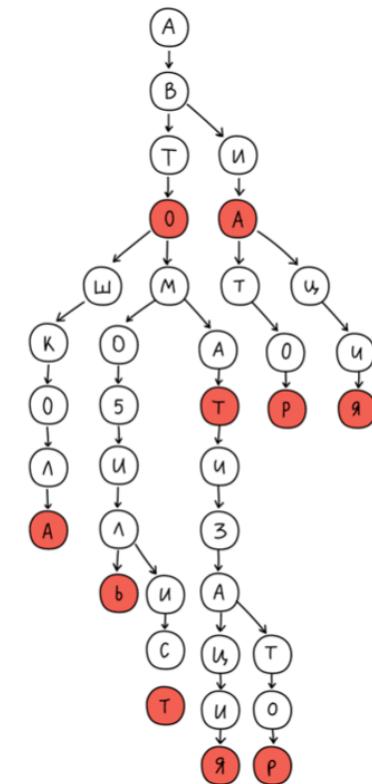
Sort-based dictionary

- Indexing time — храним токены в виде отсортированного массива или бинарного дерева поиска
- Searching time — бинарный поиск по массиву или обход дерева
- Алгоритмическая сложность поиска: $O(\log(\text{size}(\text{dictionary})))$
- Недостатки:
 - Массив: токены должны быть примерно одинаковой длины — internal fragmentation (пример: «я» и «искусственный»)
 - Дерево: перекосы

Sort-based dictionary

Предфиксное дерево поиска – trie (reTRIEval)

- Indexing time – храним токены в виде предфиксного дерева поиска
- Searching time – поиск по предфиксному дереву
- Алгоритмическая сложность поиска:
 $O(\text{length}(\text{token}))$



Hash-based dictionary

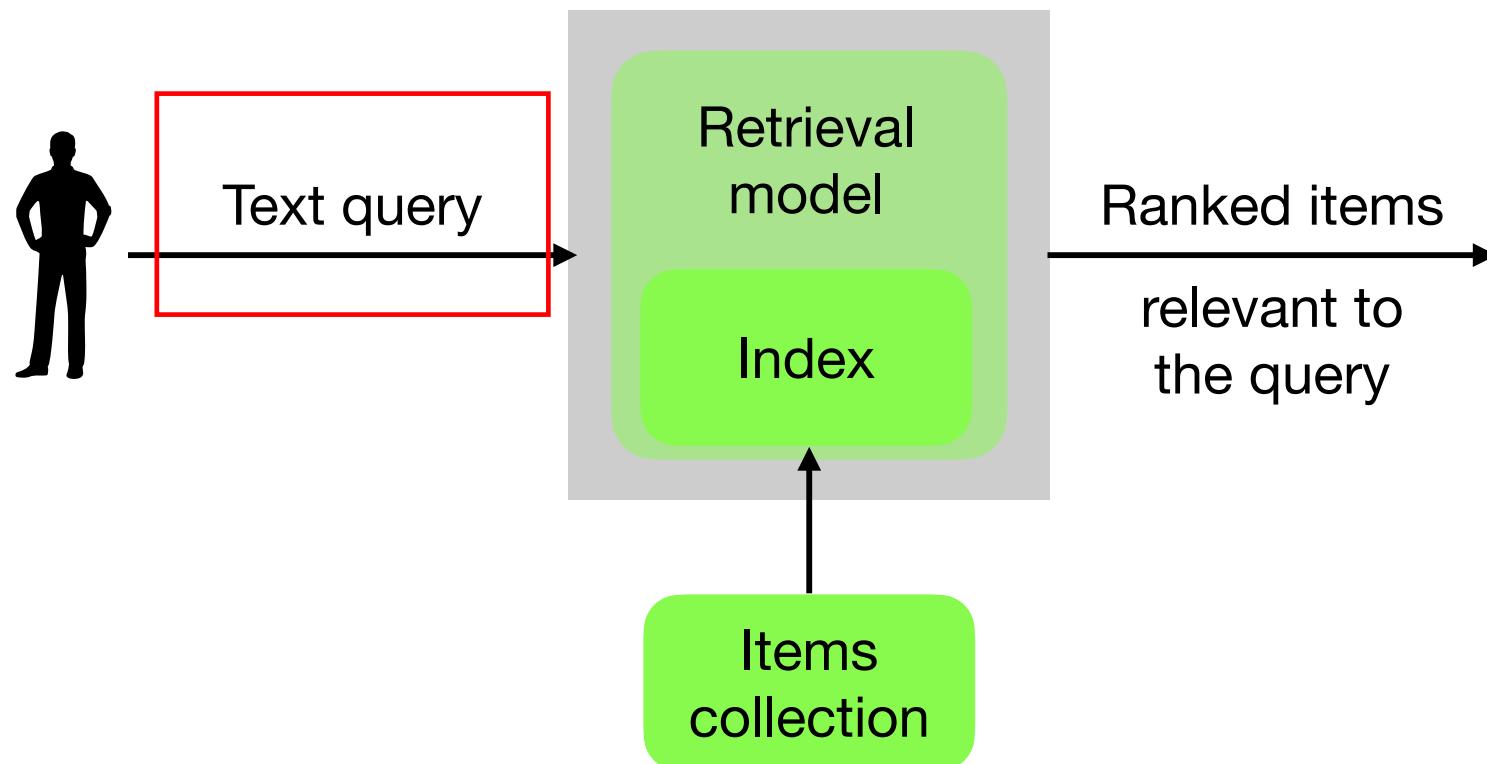
- Indexing time — хешируем каждый токен коллекции — переводим в число
 - Если мы знаем количество токенов коллекции, мы можем избежать коллизий подбором hash-функции
 - Иначе используем вспомогательные структуры данных — связный список или дерево
- Searching time — каждый токен хешируется для получения соответствующего списка документов
- Алгоритмическая сложность поиска: $O(1)$

Hash-based dictionary

- **Преимущества:**
 - Очень быстрый поиск списков документов для токенов запроса
- **Недостатки:**
 - Невозможность искать даже минимальные отклонения от токена:
«интеллект» — «интеллектуал» — «интеллектуальный»
 - Невозможность искать токены, начинающиеся с определенного префикса: «интеллек^{*}» -> [«интеллект», «интеллектуал», «интеллектуальный»]
 - Сложности при возрастающем размере словаря

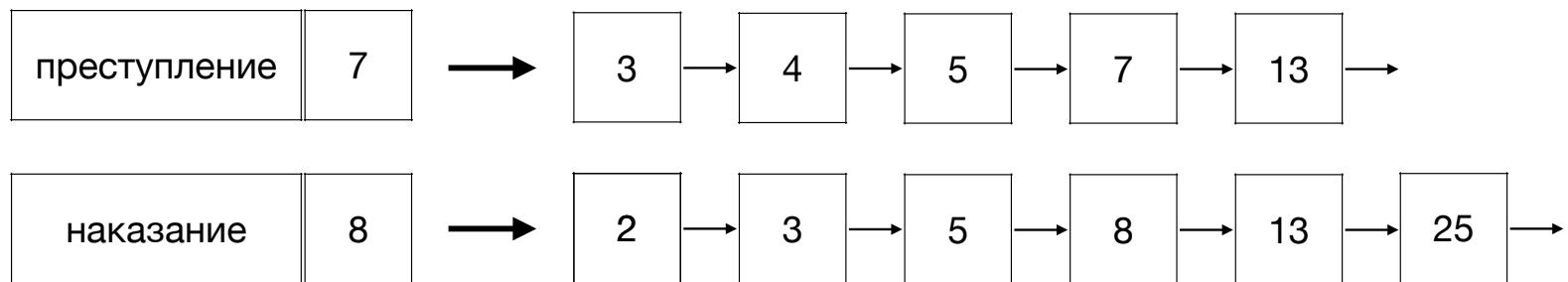
Обработка запроса

Поисковая система



Обратный индекс – term frequency

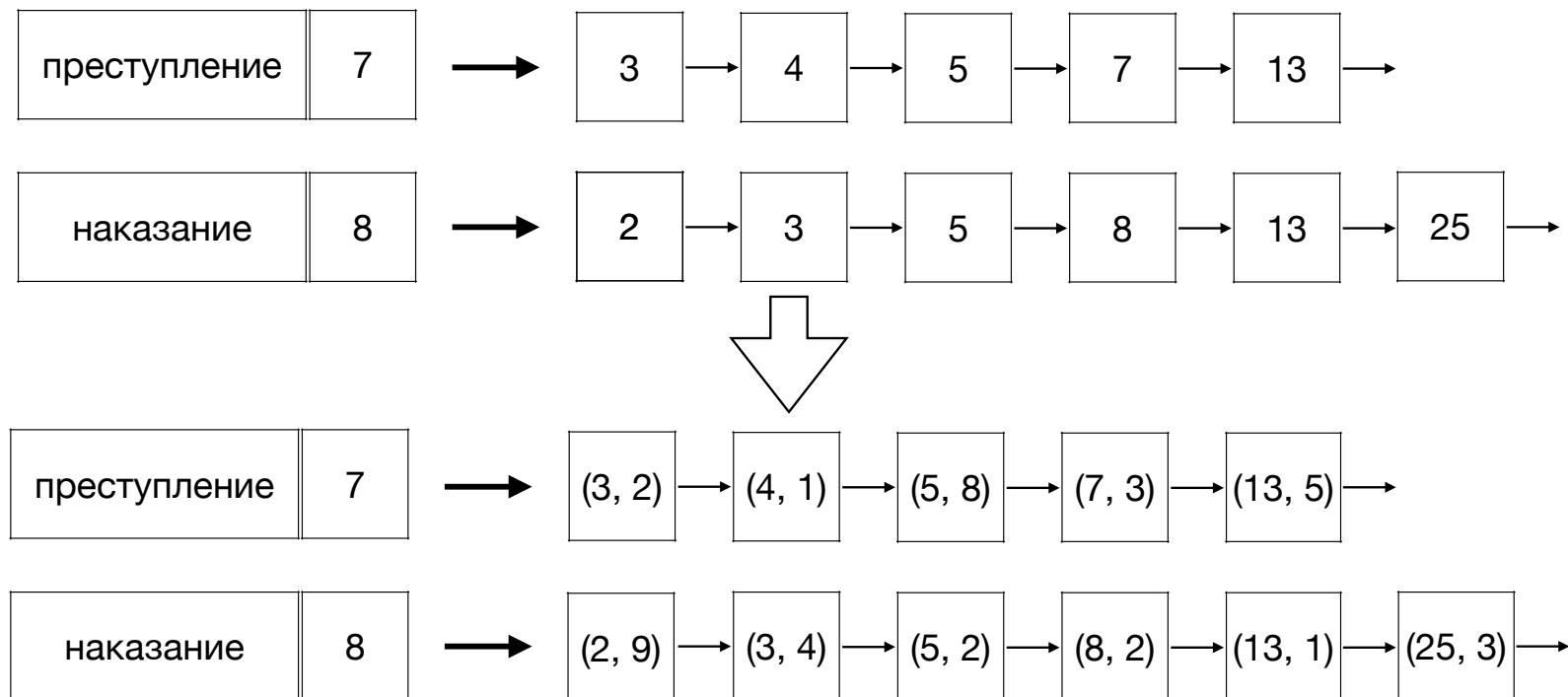
- Можно хранить еще и term frequency (tf) – количество раз, которое токен встретился в документе



- Эвристика: если токен q из запроса $Q = \{q\}$ таков, что $tf(q, d_1) > tf(q, d_2)$, то $relevance(q, d_1) > relevance(q, d_2)$

Обратный индекс – term frequency

- Можно хранить еще и term frequency (tf) – количество раз, которое токен встретился в документе



Обратный индекс – term frequency

Возможности и ограничения

- BOW-представление (нет информации о позициях токенов)
- Поддерживается булев поиск без и с ранжированием
- Поддерживаются булевые операторы AND, OR и NOT
- Поддерживается обработка запроса, причем слияние списков выполняется достаточно быстро

Обратный индекс – term frequency

Возможности и ограничения

- Булев поиск с операторами AND, OR и NOT очень ограничен
- Сложно балансировать точность (precision) и полноту (recall)

$$precision = \frac{retrieved_relevant}{retrieved}, \quad recall = \frac{retrieved_relevant}{total_relevant}$$

- «Преступление и наказание – роман Достоевского.»
- AND повышает точность, но снижает полноту
- OR повышает полноту, но снижает точность

Обратный индекс – term frequency

Возможности и ограничения

- Булев поиск с операторами AND, OR и NOT очень ограничен
- Сложно балансировать точность (precision) и полноту (recall)
- Системы булева поиска поддерживают и другие операторы:
 - **proximity** – накладывают ограничения на токены, встречающиеся близко друг к другу (с учетом порядка или без учета порядка)
 - **ограничение на поля документа** – накладывают ограничения на части документа, в которых токен может встречаться
 - **wild-card** – накладываются ограничения на матчинг токенов запроса и токенов документа

Proximity Operators

ordered window

- **A OW/N B** – означает, что токен **A** должен встретиться не более чем на **N** токенов раньше, чем токен **B**

Proximity Operators

ordered window

- **A OW/N B** – означает, что токен **A** должен встретиться не более чем на **N** токенов раньше, чем токен **B**
- Пример: «юмор OW/2 интеллект»

Proximity Operators

ordered window

- **A OW/N B** – означает, что токен **A** должен встретиться не более чем на **N** токенов раньше, чем токен **B**
- Пример: «юмор OW/2 интеллект»
 - юмор — признак интеллекта ✓

Proximity Operators

ordered window

- **A OW/N B** – означает, что токен **A** должен встретиться не более чем на **N** токенов раньше, чем токен **B**
- Пример: «юмор OW/2 интеллект»
 - юмор — признак интеллекта ✓
 - юмор — признак высокого интеллекта ✓

Proximity Operators

ordered window

- **A OW/N B** – означает, что токен **A** должен встретиться не более чем на **N** токенов раньше, чем токен **B**
- Пример: «юмор OW/2 интеллект»
 - юмор — признак интеллекта ✓
 - юмор — признак высокого интеллекта ✓
 - юмор — признак высокого и развитого интеллекта ✗

Proximity Operators

ordered window

- **A OW/N B** – означает, что токен **A** должен встретиться не более чем на **N** токенов раньше, чем токен **B**
- Пример: «юмор OW/2 интеллект»
 - юмор — признак интеллекта ✓
 - юмор — признак высокого интеллекта ✓
 - юмор — признак высокого и развитого интеллекта ✗
 - интеллект и юмор — несовместимы ✗

Proximity Operators

phrase operator

- **A B** – означает, что токен **A** должен встретиться сразу перед токеном **B**
- Эквивалентно оператору «**A OW/1 B**»
- Примеры: «пинг понг», «Нью Йорк»

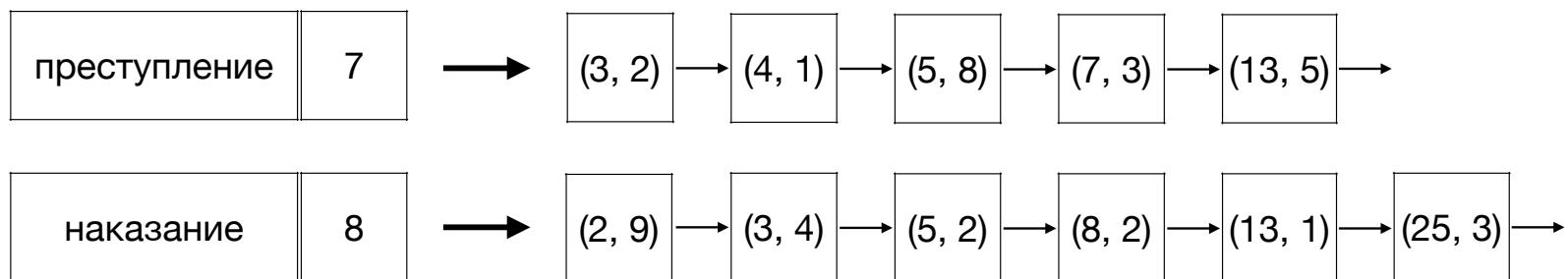
Proximity Operators

unordered window

- **A UW/N B** – означает, что токен **A** должен встретиться не дальше чем на N токенов от токена **B**
- Пример: «юмор UW/3 интеллект»
 - юмор — признак интеллекта ✓
 - юмор — признак высокого интеллекта ✓
 - юмор — признак высокого и развитого интеллекта ✗
 - интеллект и юмор — несовместимы ✓

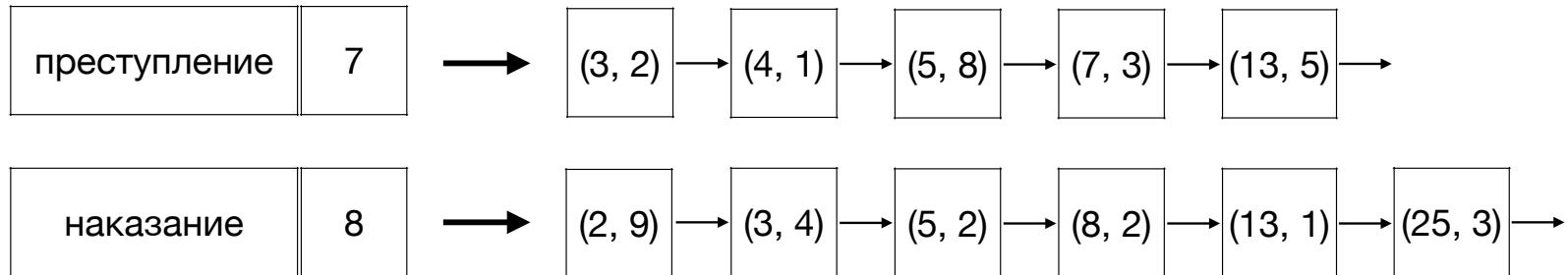
Proximity Operators

- Получится ли обработать такие операторы на построенном нами обратном индексе?



Proximity Operators

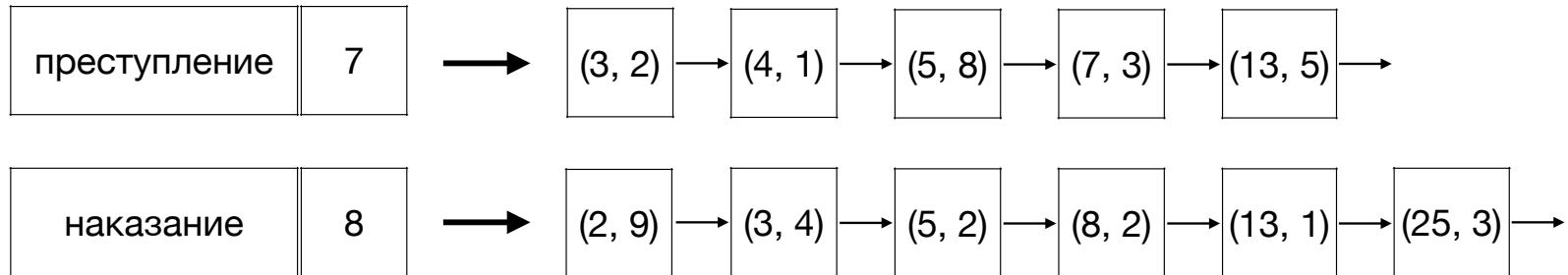
- Получится ли обработать такие операторы на построенном нами обратном индексе?



- Чего не хватает?

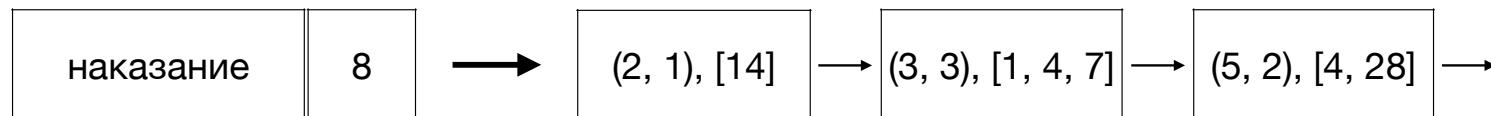
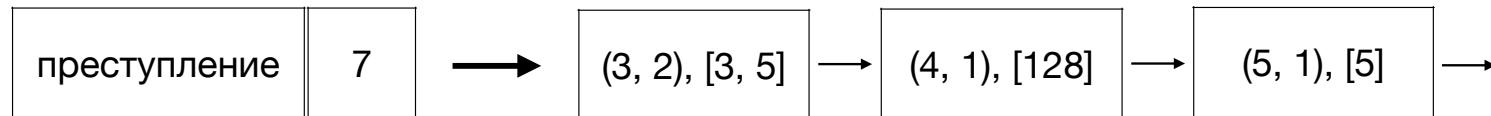
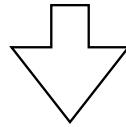
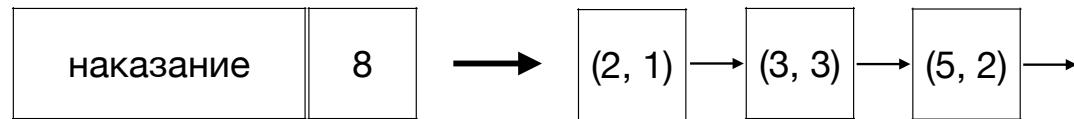
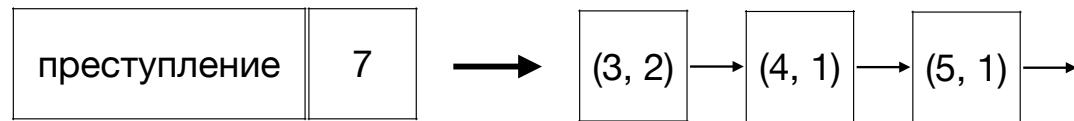
Proximity Operators

- Получится ли обработать такие операторы на построенном нами обратном индексе?



- Чего не хватает? — Токенопозиций!

Positional Index



Ограничение полей индекса

Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations

Jiaqi Zhai¹ Lucy Liao¹ Xing Liu¹ Yueming Wang¹ Rui Li¹
Xuan Cao¹ Leon Gao¹ Zhaojie Gong¹ Fangda Gu¹ Michael He¹ Yinghai Lu¹ Yu Shi¹

Abstract

Large-scale recommendation systems are characterized by their reliance on high cardinality, heterogeneous features and the need to handle tens of billions of user actions on a daily basis. Despite being trained on huge volume of data with thousands of features, most Deep Learning Recommendation Models (DLRMs) in industry fail to scale with compute. Inspired by success achieved by Transformers in language and vision domains, we revisit fundamental design choices in recommendation systems. We reformulate recommendation problems as sequential transduction tasks within a generative modeling framework (“Generative Recommenders”), and propose a new architecture, HSTU, designed for high cardinality, non-stationary streaming recommendation data. HSTU outperforms baselines over synthetic and public datasets by up to 65.8% in NDCG, and is 5.3x to 15.2x faster than FlashAttention2-based Transformers on 8192 length sequences. HSTU-based Generative Recommenders, with 1.5 trillion parameters, improve metrics in online A/B tests by 12.4% and have been deployed on multiple surfaces of a large internet platform with billions of users. More importantly, the model quality of Generative Recommenders empirically scales as a power-law of training compute across three orders of magnitude, up to GPT-3/LLaMa-2 scale, which reduces carbon footprint needed for future model developments, and further paves the way for the first foundation models in recommendations.

1. Introduction

Recommendation systems, quintessential in the realm of online content platforms and e-commerce, play a pivotal role

¹MRS, Meta AI. Correspondence to: <{jiaqiz, lucyl, xingl, yuemingw, ruii}@meta.com>. Code available at <https://github.com/facebookresearch/generative-recommenders>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

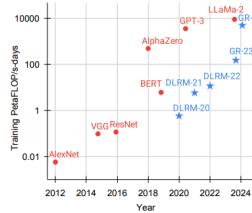


Figure 1. Total compute used to train deep learning models over the years. DLRM results are from (Mudigere et al., 2022); GRs are deployed models from this work. DLRMs/GRs are continuously trained in a streaming setting; we report compute used per year.

in personalizing billions of user experiences on a daily basis. State-of-the-art approaches in recommendations have been based on Deep Learning Recommendation Models (DLRMs) (Mudigere et al., 2022) for about a decade (Covington et al., 2016; Cheng et al., 2016; Zhou et al., 2018; Tang et al., 2020; Wang et al., 2021; Xia et al., 2023). DLRMs are characterized by their usage of heterogeneous features, such as numerical features – counters and ratios, embeddings, and categorical features such as creator ids, user ids, etc. Due to new content and products being added every minute, the feature space is of extreme high cardinality, often in the range of billions (Eksombatchai et al., 2018). To leverage tens of thousands of such features, DLRMs employ various neural networks to combine features, transform intermediate representations, and compose the final outputs.

Despite utilizing extensive human-engineered feature sets and training on vast amounts of data, most DLRMs in industry scale poorly with compute (Zhao et al., 2023). This limitation is noteworthy and remains unanswered.

Inspired by the success achieved by Transformers in language and vision, we revisit fundamental design choices in modern recommendation systems. We observe that alternative formulations at billion-user scale need to overcome three challenges. First, features in recommendation systems lack explicit structures. While sequential formulations have been explored in small-scale settings (detailed discussions

Ограничение полей индекса

Abstract

arXiv:2402.17152v3 [cs.LG] 6 May 2024

Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations

Jiaqi Zhai¹ Lucy Liao¹ Xing Liu¹ Yueming Wang¹ Rui Li¹
Xuan Cao¹ Leon Gao¹ Zhaojie Gong¹ Fangda Gu¹ Michael He¹ Yinghai Lu¹ Yu Shi¹

Abstract

Large-scale recommendation systems are characterized by their reliance on high cardinality, heterogeneous features and the need to handle tens of billions of user actions on a daily basis. Despite being trained on huge volume of data with thousands of features, most Deep Learning Recommendation Models (DLRMs) in industry fail to scale with compute. Inspired by success achieved by Transformers in language and vision domains, we revisit fundamental design choices in recommendation systems. We reformulate recommendation problems as sequential transduction tasks within a generative modeling framework ("Generative Recommenders"), and propose a new architecture, HSTU, designed for high cardinality, non-stationary streaming recommendation data. HSTU outperforms baselines over synthetic and public datasets by up to 65.8% in NDCG, and is 5.3x to 15.2x faster than FlashAttention2-based Transformers on 8192 length sequences. HSTU-based Generative Recommenders, with 1.5 trillion parameters, improve metrics in online A/B tests by 12.4% and have been deployed on multiple surfaces of a large internet platform with billions of users. More importantly, the model quality of Generative Recommenders empirically scales as a power-law of training compute across three orders of magnitude, up to GPT-3/LLaMa-2 scale, which reduces carbon footprint needed for future model developments, and further paves the way for the first foundation models in recommendations.

1. Introduction

Recommendation systems, quintessential in the realm of online content platforms and e-commerce, play a pivotal role

¹MRS, Meta AI. Correspondence to: <{jiaqiz, lucyyl, xingl, yuemingw, ruiji}@meta.com>. Code available at <https://github.com/facebookresearch/generative-recommenders>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



Figure 1. Total compute used to train deep learning models over the years. DLRM results are from (Mudigere et al., 2022); GRs are deployed models from this work. DLRMs/GRs are continuously trained in a streaming setting; we report compute used per year.

in personalizing billions of user experiences on a daily basis. State-of-the-art approaches in recommendations have been based on Deep Learning Recommendation Models (DLRMs) (Mudigere et al., 2022) for about a decade (Covington et al., 2016; Cheng et al., 2016; Zhou et al., 2018; Tang et al., 2020; Wang et al., 2021; Xia et al., 2023). DLRMs are characterized by their usage of heterogeneous features, such as numerical features – counters and ratios, embeddings, and categorical features such as creator ids, user ids, etc. Due to new content and products being added every minute, the feature space is of extreme high cardinality, often in the range of billions (Eksombatchai et al., 2018). To leverage tens of thousands of such features, DLRMs employ various neural networks to combine features, transform intermediate representations, and compose the final outputs.

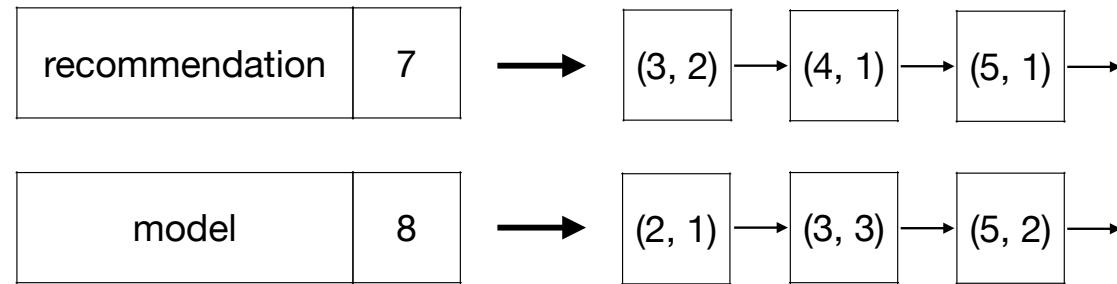
Despite utilizing extensive human-engineered feature sets and training on vast amounts of data, most DLRMs in industry scale poorly with compute (Zhao et al., 2023). This limitation is noteworthy and remains unanswered.

Inspired by the success achieved by Transformers in language and vision, we revisit fundamental design choices in modern recommendation systems. We observe that alternative formulations at billion-user scale need to overcome three challenges. First, features in recommendation systems lack explicit structures. While sequential formulations have been explored in small-scale settings (detailed discussions

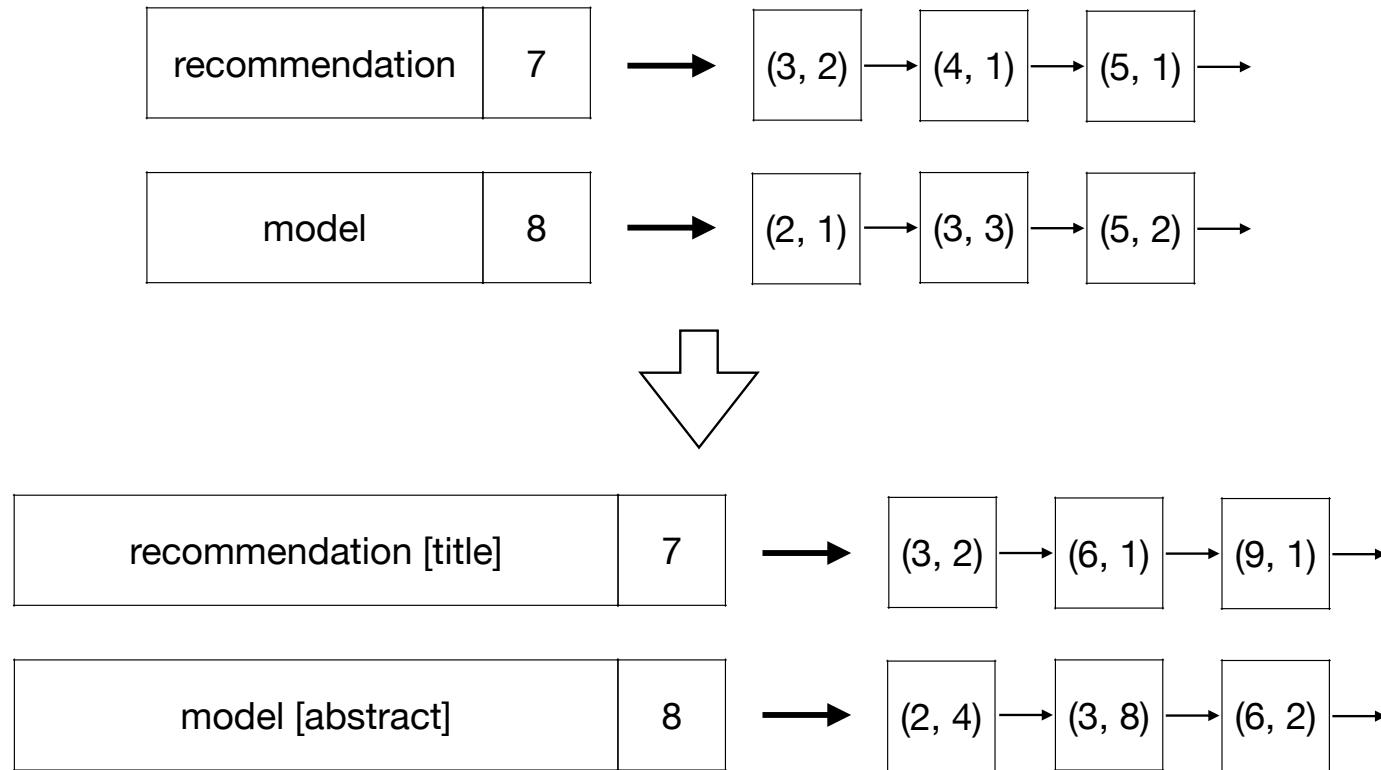
Title

Body

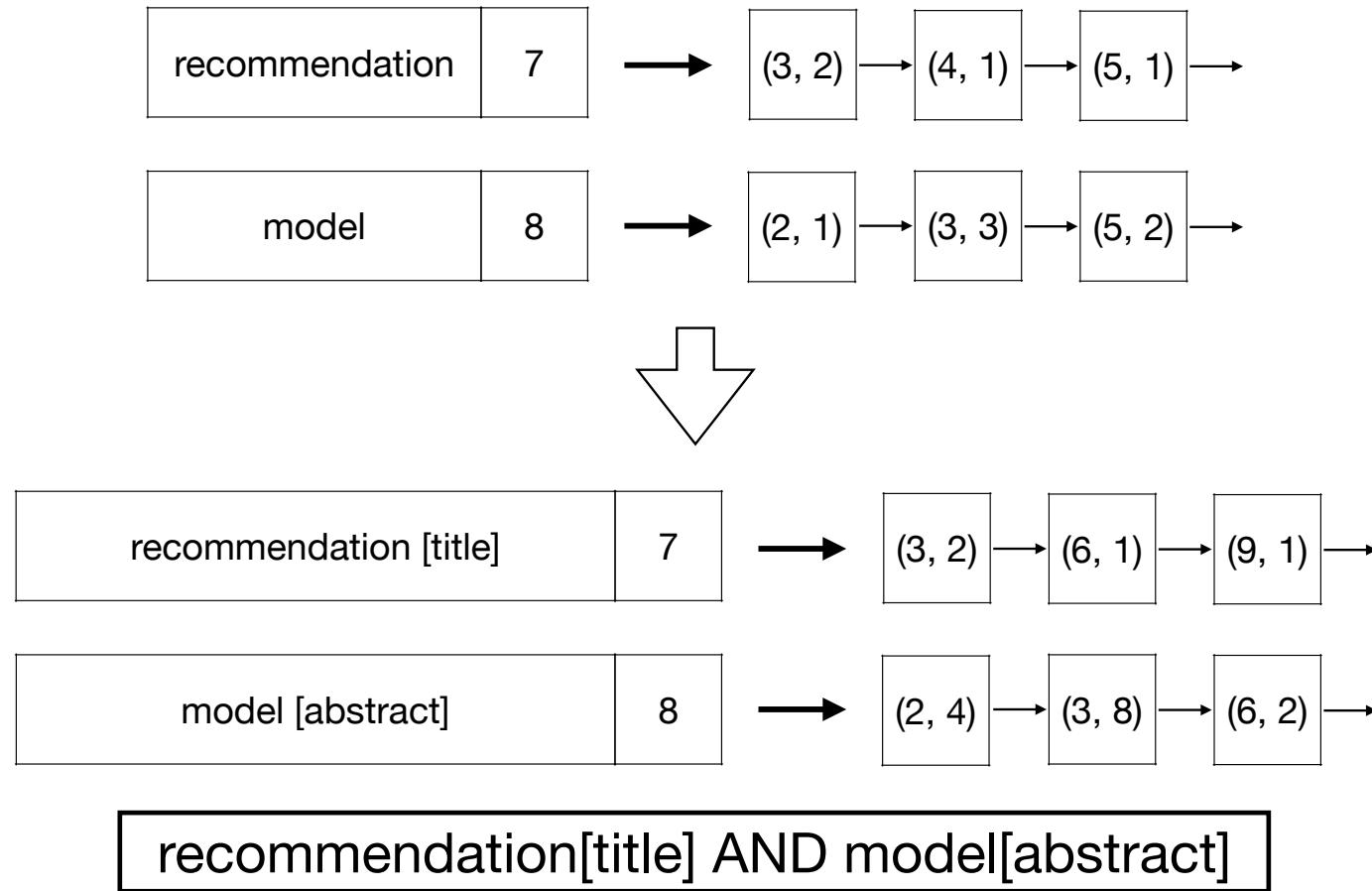
Ограничение полей индекса



Ограничение полей индекса



Ограничение полей индекса



Wild-card Operators

- Допускается матчинг одного токена запроса на несколько запросов документов
- Пример: «искусственный AND интеллек*»
- Предположение: токены [«интеллект», «интеллектуал», «интеллектуальный»] равновероятно находятся по «интеллек*»

Wild-card Operators

- Допускается матчинг одного токена запроса на несколько запросов документов
- Пример: «искусственный AND интеллек*»
- Предположение: токены [«интеллект», «интеллектуал», «интеллектуальный»] равновероятно находятся по «интеллек*»
- Как трансформировать индекс?

Wild-card Operators

- Допускается матчинг одного токена запроса на несколько токенов документов
- Пример: «искусственный AND интеллек*»
- Предположение: токены [«интеллект», «интеллектуал», «интеллектуальный»] равновероятно находятся по «интеллек*»
- Как трансформировать индекс?
- В процессе индексации также строим словарь: префикс → list[токен]
- В процессе обработки запроса ищем соответствие префикса полным токенам

Wild-card Operators

- User-query: «искусственный AND интеллек*»
- Преобразованный запрос:
«искусственный AND (интеллект **XX** интеллектуал **XX** интеллектуальный)»

Wild-card Operators

- User-query: «искусственный AND интеллек^{*}»
- Преобразованный запрос:
«искусственный AND (интеллект OR интеллектуал OR интеллектуальный)»

Wild-card Operators

- User-query: «искусственный AND интеллек^{*}»
- Преобразованный запрос:
«искусственный AND (интеллект OR интеллектуал OR интеллектуальный)»
- Аналогично с суффиксами: «^{*}лект» -> [«интеллект», «диалект»]

Wild-card Operators

- User-query: «искусственный AND интеллек^{*}»
- Преобразованный запрос:
«искусственный AND (интеллект OR интеллектуал OR интеллектуальный)»
- Аналогично с суффиксами: «^{*}лект» -> [«интеллект», «диалект»]
- Можно работать и с более сложными паттернами: «инте^{*}лект»
 - Получаем список токенов по префиксу «инте^{*}»
 - Получаем список токенов по суффиксу «^{*}лект»
 - Находим пересечение списков

Введение в информационный поиск

Андрсов Дмитрий, 09.02.2026, AI Masters