

# Information retrieval

Stanislav Protasov

2022

# Course team live in 463

**Stanislav** — s.protasov@innopolis.ru

**Anastasiia** — a.puzankova@innopolis.ru

**Marina** — m.lisnichenko@innopolis.university

**Patrik** — p.kenfack@innopolis.university

Course [news telegram channel](#)

# Agenda

1. How the course is taught and organized
  - a. Lectures and labs
  - b. Grading
  - c. Exam
2. What is “information retrieval” (IR)
  - a. Definitions
  - b. Topic overview

# How the course is taught and organized

# Major statements

Course consists of 15 weeks including **15 lectures and 15 labs.**

Course ends **in the end of April.**

**No exam.**

Course materials are in **moodle**, [\*\*github\*\*](#) and telegram.

Main **book** is “[An Introduction to Information Retrieval](#)” by Manning, Raghavan, Schütze; other materials will be published in Moodle or [referred in github](#).

# Grading and exam

- **Hometasks** (4) will cost you up to **60** points in total (15 points each)
- **Quizzes** (4): 4 short quizzes, up to **40** in total (10 point each)
- **Contests** (3-4) can bring you up to 5 additional points each.
  - +2 points for each successful completion of the task  
OR
  - +5 points for each of top-10 solutions.

## Grades distribution:

- **A = 84+**
- **B = 72-83** (rounded to integer)
- **C = 60-71**
- **Fail = 0-59**

# Information retrieval

# Definition

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

[The Book]

# Let's speculate on the definition

1. Where are borders among **algorithms, IR, and DB?**
  - a. How these disciplines answer the question  
**“How old is John Doe”?**
  - b. What is the difference in terms of software?
2. Is IR a static area?
3. Name some IR systems

# Scales of IR systems

- From **personal information retrieval**
  - Indexing vs find -r /
  - Classification (e.g. photo collection) and Filters
  - Background monitoring
- Via **enterprise and domain-specific search**
  - Specific domain information (law, chemistry, math)
  - Enterprise network (machine access)
- To **Web search**
  - Large scale
  - Commercial interest (SEO, exploits, advertisements)
  - Very heterogeneous data

# Major research milestones (1)

Early days (late 1950s to 1960s): foundation of the field

Luhn's work on automatic indexing (KWIC)

# Cleverdon's Cranfield evaluation methodology and index experiments

## Salton's early work on SMART system and experiments

1970s-1980s: a large number of retrieval models

# Vector space model

# Probabilistic models

# Major research milestones (2)

1990s: further development of retrieval models and new tasks

- Language models

- TREC evaluation

- Web search

2000s-present: more applications, especially Web search and interactions with other fields

- Learning to rank

- Scalability (e.g., MapReduce)

- Real-time search

# Highlights about today's IR

- Process **quickly** (no grep)
- **Flexible** match (consider language, typos, ...)
- Ranked retrieval (closer to query, to intent, to user, ...)
  - **Relevance** (*relevant*) - *the user perceives as containing information of value with respect to their personal information need*

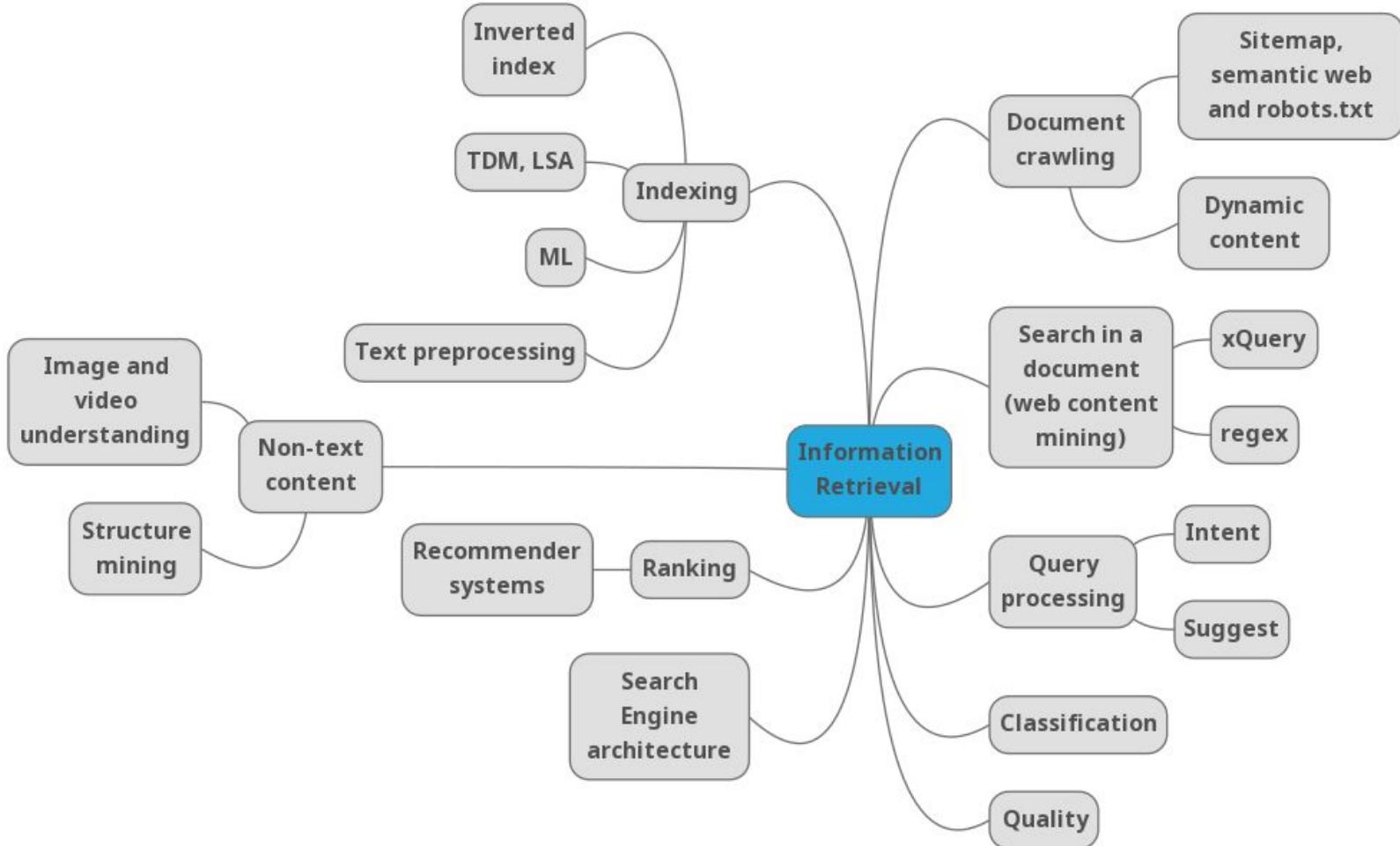
# What does IR care about?

- **Query representation**
  - Lexical gap
  - Semantic gap: ranking model vs. retrieval method
- **Document representation**
  - Specific data structure for efficient access
  - Lexical gap and semantic gap
- **Retrieval model**
  - Algorithms that find the most relevant documents for the given information need
- **Speed and space**
- ...

# IR covers ...

- Search (obviously)
- Recommendations
- Question answering
- Text mining
- Online ads
- Audio, images, video understanding
- ...

# Topic overview (by 2020)



# How search works

Watch this video: <https://youtu.be/0eKVizvYSUQ>

Answer the questions:

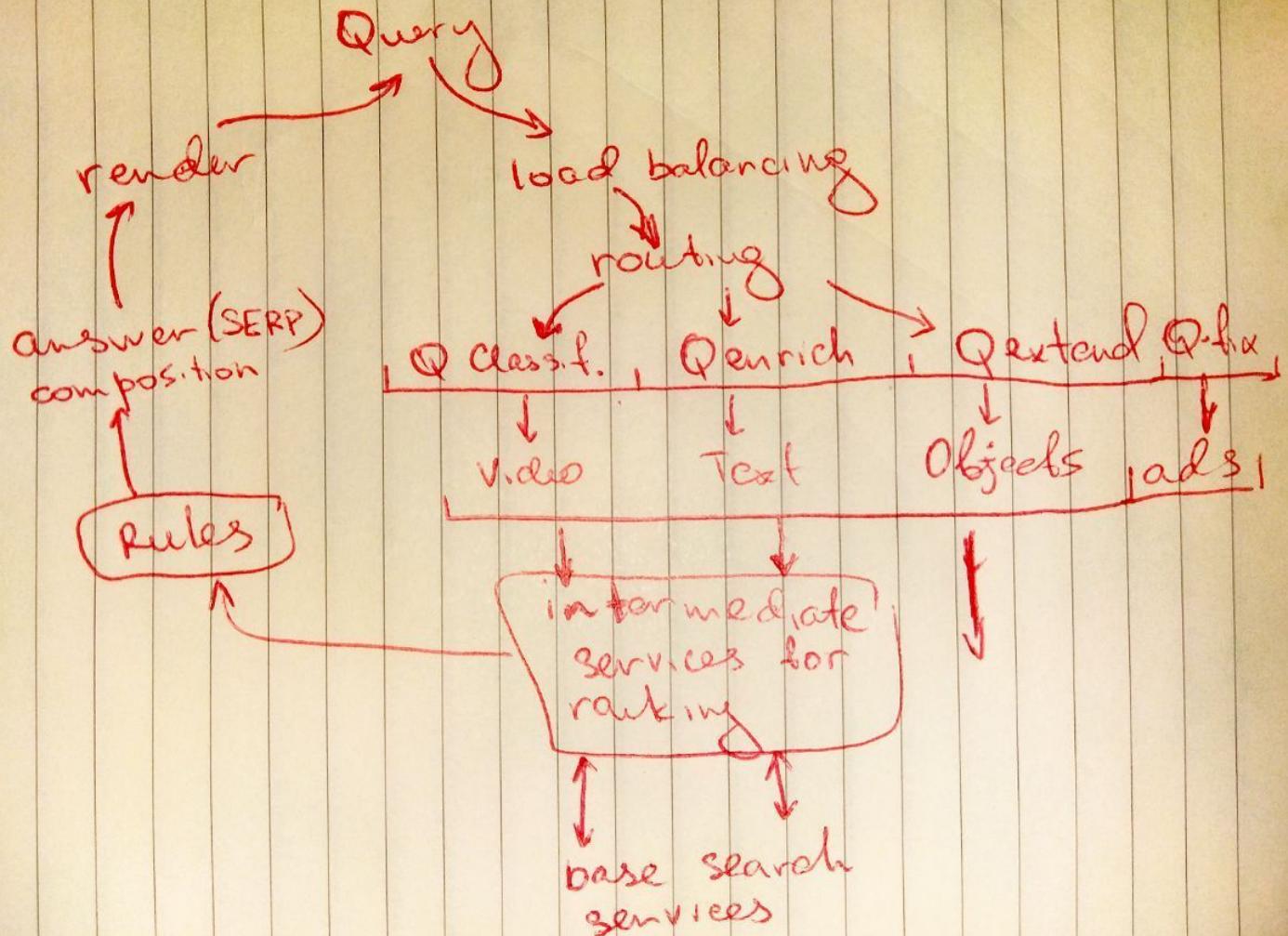
1. Did you understand how Google search works?
2. What is an **index**?
3. What is **scam** site?
4. Name or propose some **factors**
5. What is **side by side** and how is it used?

At home: read <https://www.google.com/search/howsearchworks/>

# Whiteboard time!



# Whiteboard



# Web basics

Stanislav Protasov

# Agenda

- Internet in a nutshell
- Web history and basics
  - *Idea of the Web*
  - *Web transport and formats*
  - *Browsers and DOM*
- Crawling basics
  - ***robots.txt and sitemap***
  - *Terms and Conditions*
  - *APIs*

Ok, Google, what is internet?

# IP protocol and address

213.159.212.4 (IU) - fails with browser

81.19.74.0 (LJ) - works and redirects

ping 192.34.57.61

ping 0xC022393D

ping 3223468349 # same server

http://meduza.io:@0x51134a00/

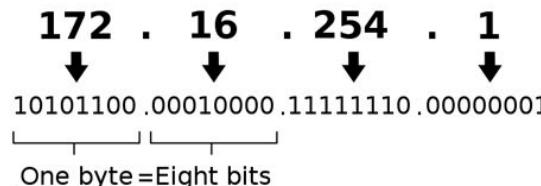
# typical phishing

IPv6 - 128 bits

Construct URL which looks like whatever, but leads to mail.ru:

1. Obtain IP address of mail.ru website
2. Convert IP to hex
3. Create `https://whatever:@ip url`

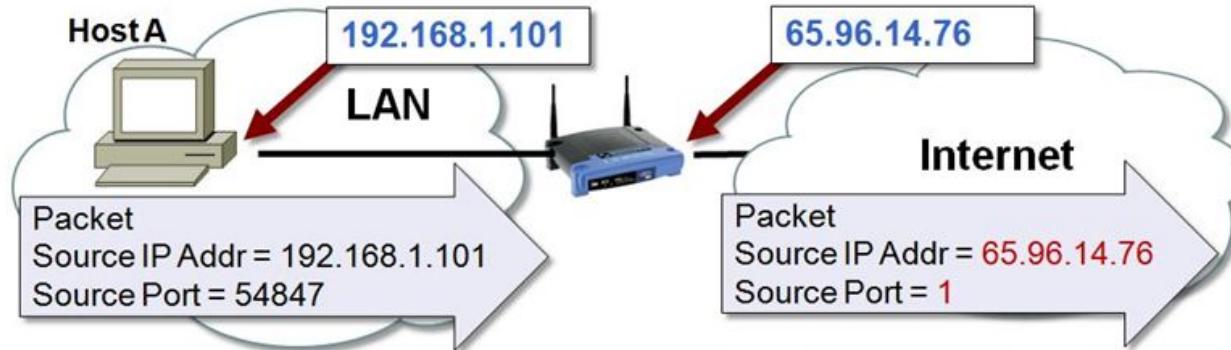
An IPv4 address (dotted-decimal notation)



Thirty-two bits (4 x 8), or 4 bytes

192.0.2.235
0xC0.0x00.0x02.0xEB
0300.0000.0002.0353
0xC00002EB
3221226219
030000001353

# IP != machine



NAT Translation Table			
Local IP Address	Source Port #	Internet IP Address	Source Port #
process X, Host A → 192.168.1.101	54,847	= 65.96.14.76	1
Host B → 192.168.1.103	24,123	= 65.96.14.76	2
process Y, Host A → 192.168.1.101	42,156	= 65.96.14.76	3
Host C → 192.168.1.102	33,543	= 65.96.14.76	4

Try on your machine

W> ipconfig.exe | findstr IPv

L/M> ifconfig | grep inet

How many addresses do you have?

# Country and IP

Using IP is one of pretty (surprisingly) reliable ways of geo location.

- [GeolP](#) 99% for country detection, 95% for city detection

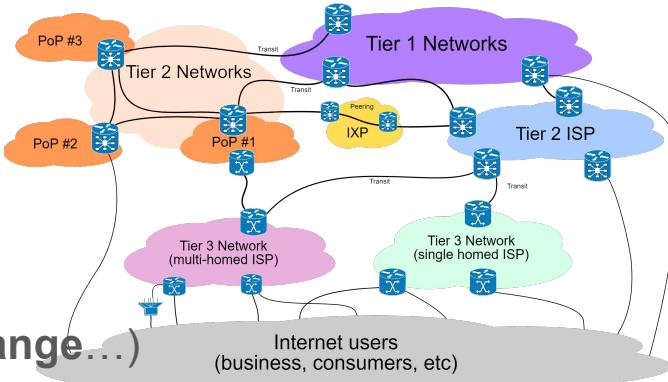
To detect country you need a **database**.

**Service processes IP of the last visible node in a chain, so:**

- NAT
- Proxy
- Turbo mode and VPN

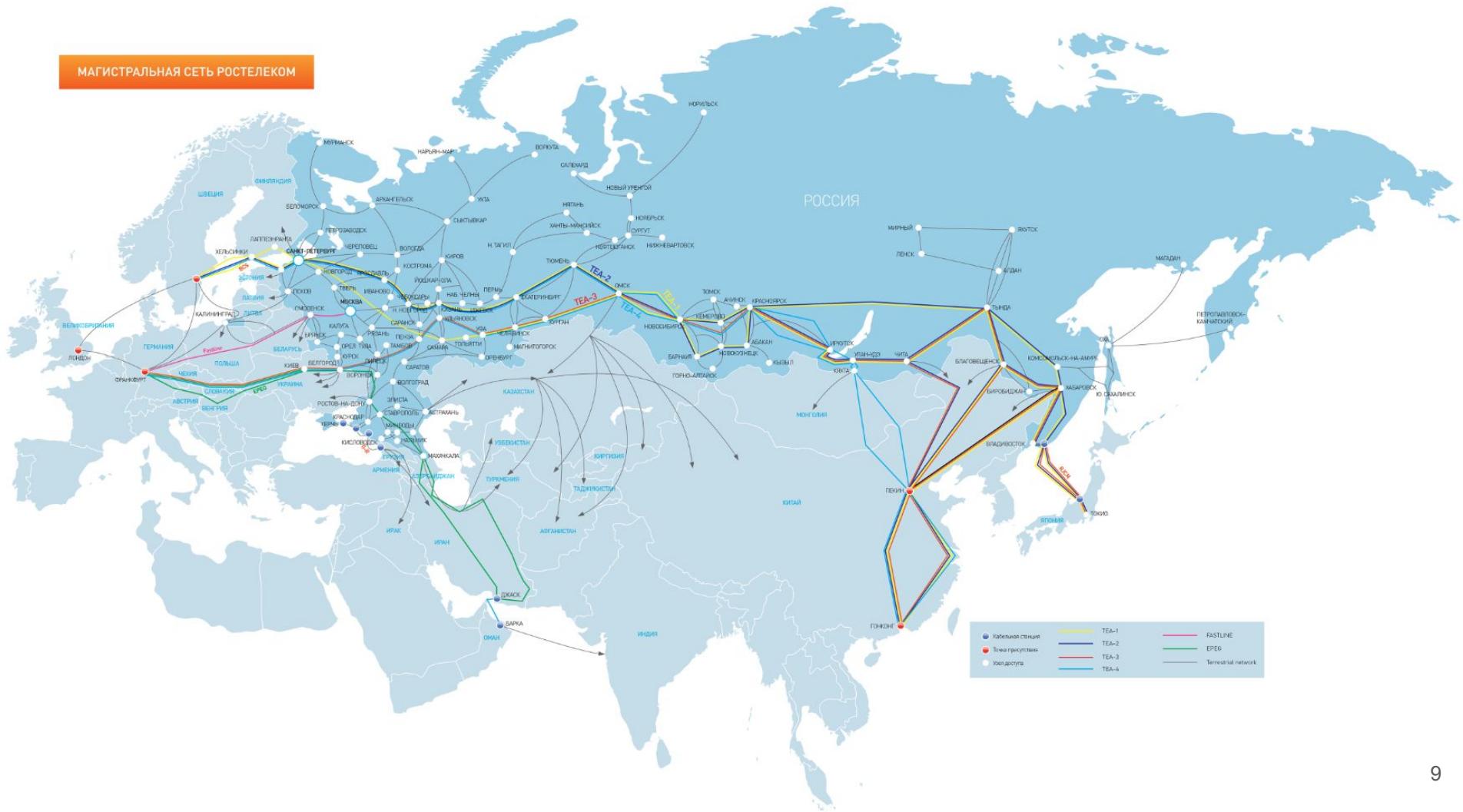
will fool the service.

# Tier-X operators (podcast in rus)



- **Tier-1 operators**
  - (e.g. Rostelecom\*, MTS\*, ... for Russia | AT&T, Orange...)
  - These networks exchange traffic (IXP) on peer conditions **for free**
  - Altogether can be considered as a **backbone** of the Internet
- **Tier-2 operators** have **partially free** peering with some segments but **paid transit** to other segments
- **Tier-3** have only paid access to the Internet
- **Takeaways:**
  - This is all about level-3 (routing) of ISO model
  - Local operators are **subjects of regulations** (RKN)
  - **Backup channels, VPNs** and so on are **important for service quality**
  - Think twice about **where to attach/rent/build your DC**
  - Small networks share the same IP-address for other networks
  - **IPv4 is over**

## МАГИСТРАЛЬНАЯ СЕТЬ РОСТЕЛЕКОМ



C:\Users\s.protasov>tracert -h 50 -w 1 rutracker.org

Трассировка маршрута к rutracker.org [195.82.146.214]  
с максимальным числом прыжков 50:

1	2 ms	1 ms	1 ms	10.91.64.1
2	22 ms	20 ms	39 ms	87.226.217.249
3	*	17 ms	18 ms	213.59.212.235
4	48 ms	48 ms	48 ms	188.254.83.102
5	*	*	*	Превышен интервал ожидания



Rutracker.org

Сканер – Раздачай!

Новости трекера

- Как тут начать
- Оценка постов
- Общие вопросы
- Что такое torrent (торрент)
- Как пользоваться BitComet
- Хочу начать!

FAQ

Как создать раздачу

Установка клиента

Установка клиентов

Как починить кеш и куки

Как пользоваться торрент-файлами

Накомпьютерную осорбисть

БитTorrent клиенты

- Несимметричные с трекером
- uTorrent
- BitComet
- Клиент под Linux
- Как настроить клиент на накомпьютерную осорбисть

FAQ

Подключение к интернету

История прогресса и фабрикации

Решение проблем с компьютером

Хеш-сумма и пакет-сылка

Кино, Видео, ТВ

- Фильмы, Наше кино, ТехноАрт
- Арт-хаус и авторское кино
- Документальные фильмы
- Дик, Фильмы, Спорт
- Аниме, Мультфильмы
- Сериалы

Книги, Изд., Уроки

- Книги, Аудиокниги
- Образование
- Из языка, Видеокурсы

Музыка, Ноты, Карaoke

- Рок музыка
- Классическая музыка
- Джаз и Блюз, Пол музыка
- Фолкмузыка

Регистрация > Вход

ПОИСК

Забыли имя или пароль?

Новости

- 31-04-20 Новый 2020 Год!!!
- 16-04-20 Новогодние конкурсы на Rutracker.org
- 01-04-20 Проект Рутрекера - Краудфандинг
- 21-03-20 Юбилейные конкурсы к 1 апреля
- 23-03-20 Встреча всех пользователей Рутрекера! Ихновы активизировались мошенники и самозванцы! Не дайте себя обмануть!

Карта форума

Последние раздачи

Товары и покупки в интернет-магазинах

AliExpress. Покупки, обзоры товаров, советы и обсуждения

ОБОХОД БЛОКИРОВОК

ОБОХОД БЛОКИРОВОК

• Блокировка IP, способы обхода и обходу • Платины для браузеров + VPN сервисы + TOR, I2P, ONION и другие распределенные сети + Обход блокировок на мобильных устройствах + Другие способы • Раздел для жалоб (недоступность Рутрекера вне РФ)

Новости

Новости трекера

- Обсуждение новостей раздачи + Авторские раздачи + Новости "Хранителей" и "Антихвара"
- Краудфандинги (переводы, покупка дисков и т. п.)
- Краудфандинги (переводы, покупка дисков и т. п.)
- Переводы: фильмы, мультфильмы, сериалы - Св Студия + Переводы: фильмы, мультфильмы, сериалы - Авторские переводчики
- Rutracker Awards (мероприятия и конкурсы)
- Конкурсы и драки + Биткоин-призы и специальных призов - Rutracker Awards (Раздачи) + Фотоконкурс. Весёлый мир на ладони

Вопросы в форуме и трекере

Правила, основные инструкции, FAQ и т.д.

Вопросы по форуму и трекеру

• Предложения по улучшению функций и трекера

Вопросы по BitTorrent сети и ее клиентам

• Проблемы со скачиванием и отдачи + Настройка антивирусов и файрволлов + uTorrent и BitTorrent + BitComet + Клиенты под OS X

Обсуждение пройдеров

• Домаги (ЭР-Телеком) + Билайн + ТТК (TransTeleKam) + Провайдеры Москвы и Московской области + Провайдеры Санкт-Петербурга и Петербургской области + Провайдеры Европы + Провайдеры Беларусь + Ростелеком (региона)

Жалобы, конфликты и переговоры

• Жалобы: коммерческие проблемы + Подбор конфигурации, выбор и обсуждение контент-провайдеров + Сетевое оборудование + Выбор и обсуждение антивируса + Мобильные устройства

Кино, Видео и ТВ

Предложения по улучшению категорий "Кино, Видео и ТВ"

Кино, Видео и ТВ - помощь по раздаче

• Заводы, закачки, координации + Обработка видео + аудио + Работа с DVD + Работа с BLU RAY и HD-DVD

Наши кино

C:\Users\s.protasov>tracert -h 50 -w 1 rutracker.org

Трассировка маршрута к rutracker.org [195.82.146.214]  
с максимальным числом прыжков 50:

1	*	*	3 ms	192.168.43.116
2	*	*	*	Превышен интервал ожидания для запроса.
3	*	*	*	Превышен интервал ожидания для запроса.
4	*	*	*	Превышен интервал ожидания для запроса.
5	*	*	*	Превышен интервал ожидания для запроса.
6	*	*	*	Превышен интервал ожидания для запроса.
7	*	*	*	Превышен интервал ожидания для запроса.
8	57 ms	38 ms	47 ms	37.29.5.65 Megafon
9	*	*	*	Превышен интервал ожидания для запроса.
10	*	*	*	Превышен интервал ожидания для запроса.
11	*	*	*	Превышен интервал ожидания для запроса.

Антивирус ESET NOD32

Защищите любые 3 устройства

Стоимость — 5 ₽ / день

Подключить

Отменить

Нажимая «Подключить», вы соглашаетесь с условиями услуги «Антивирус ESET NOD32»

Доступ к информационному  
ресурсу ограничен  
на основании Федерального  
закона от 27 июля 2006 г.  
№ 149-ФЗ «Об информации,  
информационных  
технологиях  
и о защите информации».

# Try on your machine

W: **tracert -w 1 ya.ru**

L/M: **traceroute -w 1 ya.ru**

\*\* sudo apt-get install traceroute

also not sure about Linux Subsystem for Windows

## Where is provider, where is Yandex?

```
C:\Users\stani>tracert -w 1 ya.ru
```

Tracing route to ya.ru [87.250.250.242]  
over a maximum of 30 hops:

**IU corporate network**

1	1 ms	1 ms	1 ms	10.91.64.1	Tattelekom
2	4 ms	4 ms	4 ms	1.123.18.84.in-addr.arpa [84.18.123.1]	RETN (T2)
3	3 ms	3 ms	3 ms	ae11-498.rt.itp.kzn.ru.retn.net [87.245.231.208]	
4	18 ms	17 ms	19 ms	ae3-4.rt.m9.msk.ru.retn.net [87.245.233.89]	
5	*	59 ms	61 ms	gw-yandex.retn.net [87.245.229.253]	MSK-IX
6	79 ms	66 ms	63 ms	sas-32z5-ae2-1.yndx.net [87.250.239.203]	
7	*	*	*	Request timed out.	Sasovo Yandex DC
8	62 ms	65 ms	62 ms	ya.ru [87.250.250.242]	

# Name services

**DNS — domain name system**, allows to operate human-readable names instead of addresses

There are 13 core [Root Servers](#) ([a..m].root-servers.net) responsible for the Internet. Lower level responsible for domains, subdomains, ...

DNS supports forward (domain → IP) and reverse (IP → domain) requests

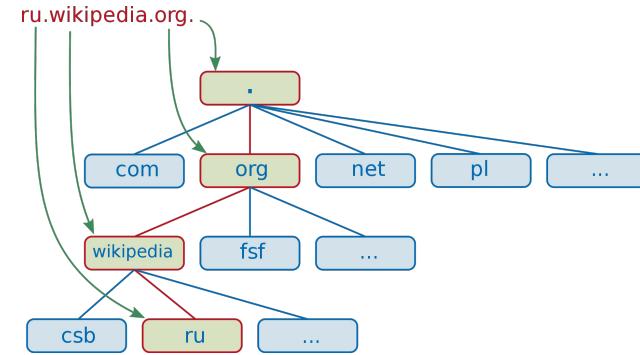
**nslookup sprotasov.ru**

**nslookup 192.34.57.61**

**nslookup code-test.ru**

```
root@simpletrack:~# ping -c 1 yandex.ru
PING yandex.ru (77.88.55.80) 56(84) bytes of data.
64 bytes from yandex.ru (77.88.55.80): icmp_seq=1 ttl=244 time=116 ms

--- yandex.ru ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 116.697/116.697/116.697/0.000 ms
root@simpletrack:~# logout
Connection to sprotasov.ru closed.
(base) stranger@sprotasovn:~$ ping yandex.ru
PING yandex.ru (213.180.193.56) 56(84) bytes of data.
64 bytes from familysearch.yandex.ru (213.180.193.56): icmp_seq=1 ttl=56
```



# Resource record

The screenshot shows the DigitalOcean Control Panel interface for managing DNS records. The left sidebar includes sections for PROJECTS (with 'stanislav.protasov' selected), MANAGE, DISCOVER, and ACCOUNT. The main content area is titled 'DNS records' and displays a table of existing records.

**DNS records table:**

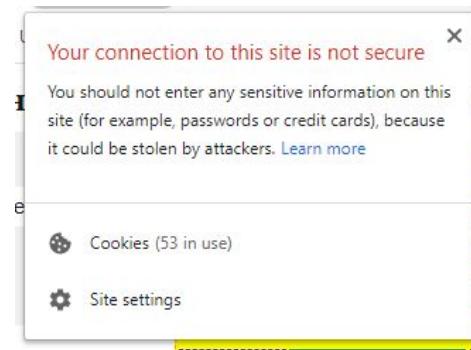
Type	Hostname	Value	TTL (seconds)	More
NS	sprotasov.ru	directs to ns3.digitalocean.com.	1800	More ▾
A	sprotasov.ru	directs to 192.34.57.61	1800	More ▾
NS	sprotasov.ru	directs to ns1.digitalocean.com.	1800	More ▾
NS	sprotasov.ru	directs to ns2.digitalocean.com.	1800	More ▾

**Create New Record Form:**

HOSTNAME: Enter @ or hostname  WILL DIRECT TO: Select resource or enter custom IP  TTL (SECONDS): Enter TTL 3600  Create Record

# Short quiz. Test yourself

1. One website opens for me, but another — does not. Same for my neighbour in the office. Why?
2. One website opens for me, but another — does not. Both work for my neighbour in office. Why?
3. I clicked a link starting with *google.com* in my email. I know that all google pages are secured, but browser says that Why?



# Before we continue - Internet

1. **Internet and IP.** IP manages (routes) how data is flowing from one machine (e.g. server) to another (e.g. smartphone). Providers are working on this level
  - a. Tools: tracert, ping, ifconfig/ipconfig
  - b. **TCP** manages how to transfer more than one packet of date preserving order and integrity
    - i. Tools: nc, telnet
2. **DNS.** Assigning string names to IP-addresses allow to establish many-to-many relations, thus, make infrastructure faster and reliable. Domain names are objects of legal regulations ([whois](#))
  - a. Tools: nslookup, ping

# Web and HTTP

# ~~KEY~~ BUZZ WORDS

**Internet** – network for transferring information among devices

**WWW (Web)** – graph of documents (hypertext), placed at web-servers, that are connected to Internet

**Hypertext** – text, that contain references to other texts

**HTML** (hypertext markup language) – standard of hypertext for Internet

**Web 2.0** – everything, that is beyond static HTML documents: social networks, blogs, video-hostings, internet-marketing, web application and services. Service-oriented network.

**Web 3.0 (*semantic web*)** – graph of machine-readable, semantically rich documents. Content-oriented network.

# HYPertext Evolution

Web 1.0	Web 2.0	Web 2.0, 3.0
<b>HTML</b> – Subset of SGML (markup language)	<b>xHTML</b> – fusion of HTML tags and XML standard	<b>[x]HTML5</b> – valuable layout changes, semantic tags were added
SGML parsers	XML parsers	<b>HTML5/XML</b> parsers
For presentation in browsers	For displaying interactive and media content	For creation of web-applications that support semantic markup

# URI VS URL

**Uniform resource identifier (**URI**)** – machine-readable text identifier of the resource, created according specific rules

URI common syntax: *scheme:scheme-specific-part*

**Uniform resource locator (**URL**)** – subset of URI, describes location and way (protocol) to access object in the Internet

`http://www.mail.ru/`

**Uniform resource name (**URN**)** – subset of URI, identifies object, but does not locate it

`urn:isbn:0451450523` or

Magnet links: `magnet:?xt=urn:btih:c12fe1c06bba254a9dc9f519b335aa7c1367a88a`

# URL SYNTAX



Examples:

- <https://mail.google.com/>
- <ftp://root:qwerty@ftp.example.com/>
- <wss://server.name:443/method/name>
- <http://sprotasov.ru/index.html#author:Alexandr%20Buyanov>
- <https://innopolis.com/index.php/fdgdflgkdjf;lgk%20djf;lgkdfj;gl%20kfdj;gldkfgj%20;d1kgdj;g%20lkdjg%20;df>

# HTTP

**HTTP** (hypertext transfer protocol) – application (7) level protocol to deliver text data. Created to transfer hypertext. Provide communication between *client* (usually browser) and *server* (web-server) using client requests and server responses.

**HTTP v1.0** – does not support using single TCP session for multiple requests. Supports following client request methods:

- GET – get content from the server
- HEAD – get only header from the server without content ("what to expect")
- POST – sent data to the server

**HTTP v1.1.** – supports also PUT, DELETE, TRACE, OPTIONS, CONNECT, PATCH

**HTTP/2** - SDPY (Google) based update. Binary. Header compression, Server pushes, conveyor requests, request multiplexing over single TCP

# HTTP HEADER

HTTP request =

Request URI + **HOST** + [[headers]] + <empty\_line> + **body**

```
POST /index.html HTTP/1.1
HOST: example.org:8080
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml
```

...

param1=value1&param2=value2

HTTP response=

Response code + **headers** + <empty\_line> + **body**

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 3123
...
```

<html>...

# Demo with telnet

o sprotasov.ru 80

HEAD / HTTP/1.1

host:sprotasov.ru

HEAD / HTTP/1.1

host:code-test.ru

GET / HTTP/1.1

host:code-test.ru

# IMPORTANT HEADERS

## REQUEST

Accept, Accept-Charset, Accept-Encoding – formats, that your browser understands (text/plain, application/xml), encodings (utf-8) and supported compression algorithms (gzip, deflate)

Authorization – header that stores authentication type, credentials/keys,...

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

Content-Length – request body length (same for response)

User-Agent – browser and operating system

Upgrade – request to change communication protocol (Upgrade:websocket)

## RESPONSE

Cache-Control – time to store document in a browser cache

Content-Encoding, Content-Language, Content-Type – content characteristics

# Browser fingerprint: domain I see first time

The screenshot shows a browser window with the title "Example Domain". The address bar indicates "Not secure | example.com". The developer tools Network tab is active, showing a timeline from 100 ms to 1000 ms. A single request to "example.com" is listed, showing a duration of approximately 300 ms. The request details pane shows the following headers:

Name	Value
Host	example.com
Connection	keep-alive
Upgrade-Insecure-Requests	1
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.116 Safari/537.36
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

At the bottom of the developer tools, the "What's New" tab is selected.

# RESPONSE CODES

- 1xx – information
  - 101 - Switching Protocols
- 2xx – success
  - 200 – OK
  - 201 – created (new resource)
  - 206 – partial content
- 3xx – redirection (specified header Location: addr)
  - 301 – moved permanently
  - 304 – not modified
- 4xx – client-side error
  - 401 – unauthorized
  - 404 – not found
- 5xx – server-side error
  - 503 – server unavailable

# REQUESTS AND SYSTEM STATE: SAFE AND IDEMPOTENT REQUESTS

Safe request does not change server/object state. For getting some information

Idempotent request if you make 2 or more identical requests, second and other requests do not change server/object state

$F(state) == F(F(state))$

	SAFE	NOT SAFE
IMDEMPOENT	HEAD, GET, OPTIONS, TRACE	PUT, DELETE
NON-IMDEMPOENT	--	POST, PATCH

# PARAMETERS

GET params:

- `http://server.name/path?param1=value1&param2=value2`

POST params:

- POST /path HTTP/1.1

`param1=value1&param2=value2`

# COOKIES

- Cookies – small **drive space** to store data sent by server to browser. Max – 4KB
- We need cookies for **stateful services** (e-shop cart, etc) or for storing session keys
- Cookies have life period and are **sent** to server with **each request**

# COOKIES EXAMPLES

```
GET /index.html HTTP/1.1  
Host: www.example.org
```

browser -----> server

```
HTTP/1.0 200 OK  
Content-type: text/html  
Set-Cookie: name=value  
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT
```

(content of page)

browser -----<----- server

```
GET /spec.html HTTP/1.1  
Host: www.example.org  
Cookie: name=value; name2=value2  
Accept: */*
```

browser -----> server

# Before we continue - Web

1. **Hypertext, HTML and HTTP.** Hypertext is an approach to represent **linked** documents (altogether = The Web). xHTML5 is a de-facto standard. HTTP - a protocol for transferring [hyper]text data, or text-encoded media (base64). Defines methods (GET, POST, ...), status codes (200, 403, 502), headers (metainformation), sessions (1.1+). Works over TCP (means one HTTP message can be bigger than 1 IP frame).
  - a. Tools: telnet, curl, wget, postman
2. **URI ⊃ URL.** URL is a standard way to define together:
  - a. Where is the document (domain + port + path)
  - b. How to access the document (protocol, credentials)

# FTP

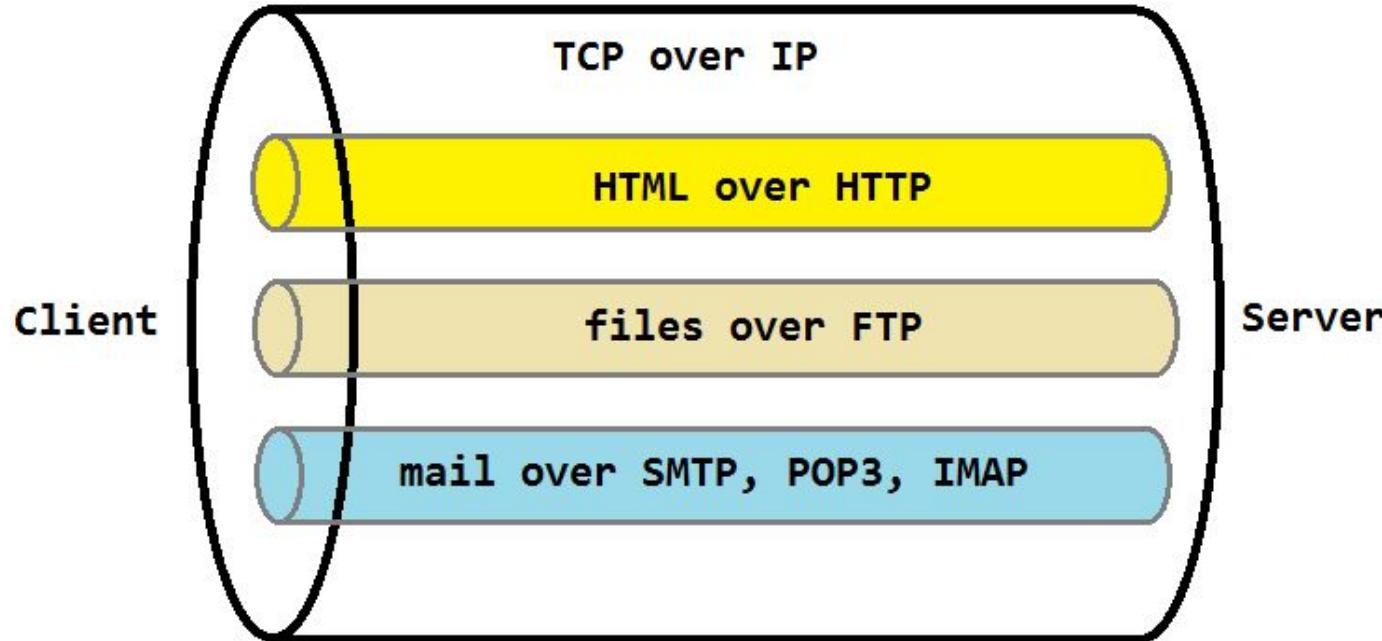
**FTP (file transfer protocol)**, 1971. Consists of greeting session and request session (VERB param [params]\015\012), and server responses

<b>Client:</b>		Connects @server:21
<b>Server:</b>	220 Hello,...	
<b>Client:</b>	USER MB1234	
<b>Server:</b>	331 Password required to access user account MB1234.	
<b>Client:</b>	PASS QXJ4Z2AF	PLAIN TEXT
<b>Server:</b>	230 Logged in.	
<b>Client:</b>	CWD Bills	Change directory to "Bills."
<b>Server:</b>	250 "/home/MB1234/Bills" is new working directory.	
<b>Client:</b>	PORT 192,168,1,2,7,138	accepts data @client:1930 [=7*256 + 138]
<b>Server:</b>	200 PORT command successful.	
<b>Client:</b>	LIST	Send the list of files in "Bills."
<b>Server:</b>	150 Opening ASCII mode data connection for /bin/ls.	server connects out from its port 20 to port client:1930
<b>Server:</b>	226 Listing completed.	succeeded
<b>Client:</b>	PORT 192,168,1,2,7,139	
<b>Server:</b>	200 PORT command successful.	
<b>Client:</b>	RETR Yoyodyne.TXT	Download "Yoyodyne.TXT."
<b>Server:</b>	150 Opening ASCII mode data connection for Yoyodyne.TXT	
<b>Server:</b>	226 Transfer completed.	succeeded
<b>Client:</b>	QUIT	
<b>Server:</b>	221 Goodbye.	

# EMAIL: SMTP, IMAP, POP3

- **SMTP** (simple mail transfer protocol, @:25) – for **transferring messages between servers** and for **server-client** communication. FTP's brother.
- **POP3** (post office protocol v3, @:110) – standard protocol for **client to get messages from server**
- **IMAP** (internet message access protocol, @:143) – **standard protocol for client to get messages from server**; has **sending implementation** (considered bad), keeps session, supports multiple clients for 1 mailbox.

# OVERVIEW



# Web security (client side)

# TERMS

**Identification** – assigning labels (IDs) to objects, as long as process of comparing one label with the list

**Authentication** – procedure of checking authenticity, proving match between ID and object. We can authenticate user (ID + password), machine, document (digital signature). Can be multi-factor, one-way, both-way

**Authorization** – granting access to perform some action

# FUNCTION OF HTTP-AUTHENTICATION

- Limiting access by means of HTTP protocol
  - Rare for sites. Most sites use forms-based authentication
  - Common for **services** and **APIs**. (access not via browser UI, but server or ajax code)

# COMMON FACTS ABOUT AUTHENTICATION

If server returns 401, this means it wants to authenticate you. Server must send WWW-Authenticate header to you.

HTTP/1.0 **401** Unauthorized

Cache-Control: no-cache

Pragma: no-cache

Content-Length: 58

Content-Type: text/html

Expires: -1

Server: Microsoft-IIS/8.0

**WWW-Authenticate: Basic realm="area to be accessed"**

# BASIC AUTHENTICATION

Easiest way to setup authentication

```
GET /sometail.aspx HTTP/1.1  
Host: somehost  
Authorization: Basic bG9naW46cGFzc3cwcmQ=
```

where

“**bG9naW46cGFzc3cwcmQ=**” == base64(“**login:passw0rd**”)

NB:

- Login and password are not secured in fact! Only way to use – over HTTPS
- You can send this without challenge
- With each request

# DIGEST AUTHENTICATION

HTTP/1.1 401 Unauthorized

WWW-Authenticate: Digest realm="testrealm@host.com",  
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",  
opaque="5ccc069c403ebaf9f0171e9517f40e41"

....

Authorization: Digest username="Mufasa",  
realm="testrealm@host.com",  
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",  
uri="/dir/index.html",  
response="e966c932a9242554e42c8ee200cec7f6",  
opaque="5ccc069c403ebaf9f0171e9517f40e41"

HA1 =  $\overline{MD5(A1)} = MD5(username : realm : password)$   
HA2 =  $MD5(A2) = MD5(method : digestURI)$   
response =  $MD5(HA1 : nonce : HA2)$

RFC 2617:

$$\begin{aligned} HA1 &= MD5(A1) = MD5(username : realm : password) \\ HA2 &= MD5(A2) = MD5(method : digestURI) \end{aligned}$$

Если значение директивы QOP равно «auth-int», то HA2 равняется:

$$HA2 = MD5(A2) = MD5(method : digestURI : MD5(entityBody))$$

Если значение директивы QOP равно «auth» или «auth-int»,

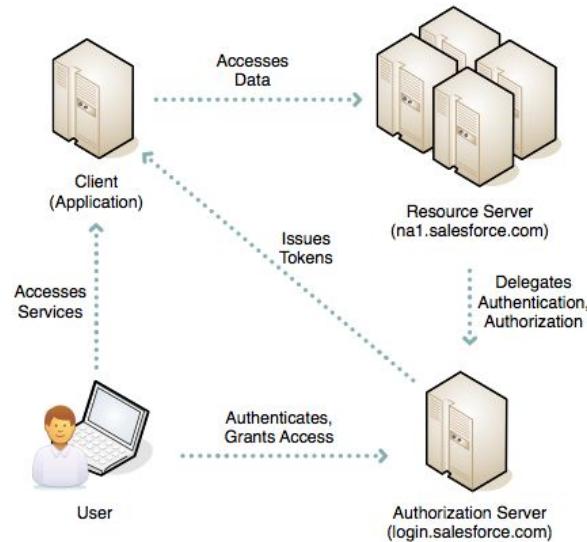
$$response = MD5(HA1 : nonce : nonceCount : clientNonce : qop : HA2)$$

Если директива QOP не определена, то ответ вычисляется так:

$$response = MD5(HA1 : nonce : HA2)$$

# OAUTH

- Authenticates application on behalf of user (or anonymously)
- Based on Access Tokens. Twitter example



# FORMS AUTHENTICATION

- Not a part of HTTP protocol
- Based on HTML <FORM>-tag and request parameters mechanism

```
request destination  
|  
<form action="Default.aspx" method="get">  
    Login: <input type="text" name="username" />  
    <br/><br/>  
    Password: <input type="password" name="password" />  
    <br/><br/>  
    <input type="submit" value="Log me in"/>  
</form>
```

request method (get/post)

"username=..." param will be created

"password=..." param will be created

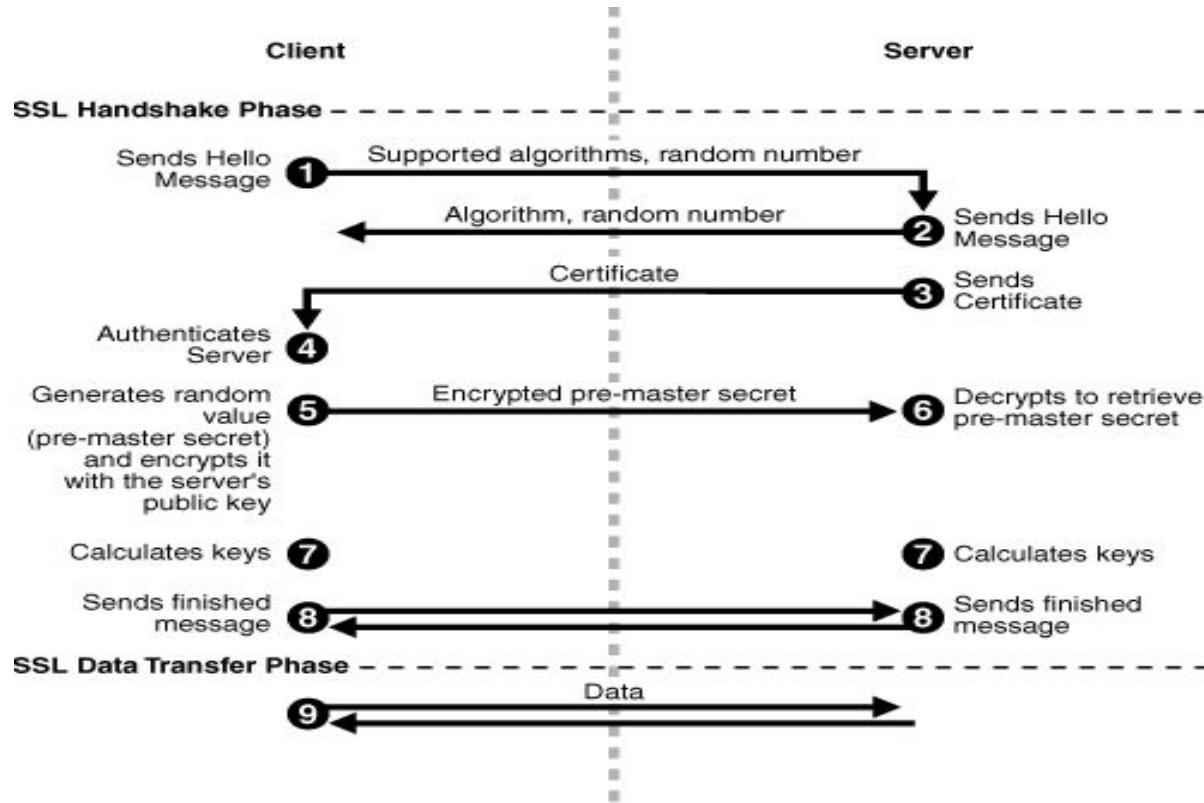
covers field with "stars"

button that triggers submission (request)

# HTTP SECURE

- **HTTPS** = HTTP over SSL/TLS
  - SSL – protocol with asymmetric cryptography and symmetric encoding
  - TLS = SSL v3
- **HTTP (FTP, telnet)** work transparently over SSL/TLS
  - Firstly client's application (browser) performs “handshake”.
  - Then the channel is created and data is sent over this channel using standard protocol (e.g. HTTP)

# HTTPS HANDSHAKE



# CERTIFICATES

**Digital certificate** – electronic document (file), ensuring that public key belongs to bearer. Certificate must be signed by certification authority.

- Mandatory cert parts:
  - resource ID (**Subject**)
  - **public** key
  - certification authority (**Issuer**)
- Optional cert parts
  - **private key**
  - usage restrictions

# Certificates examples

<https://tv.eurosport.com/>

W: certmgr.msc

# Before we continue - Security

1. HTTP supports **Basic and Digest authentication** of a user from the box (defined in standard). Mostly used for service-to-service interaction. **Data is still plaintext.**
2. **OAuth** is a new way to grant access to the service. Access to APPLICATION on behalf of a user. 3-sided:
  - a. **User** passes login/pass to **authentication service**
  - b. **Auth service** issues a token for **an app** to act on behalf of a user.
  - c. **Application** uses token to interact with **a service**.

**Data is still plaintext.**

3. **TLS/SSL** is used to establish **secure channel over TCP**. Uses asymmetrics keys to build end-to-end encrypted communication (session save overhead!). Certificates are used as containers for keys + validation tool. HTTP over SSL = <https://....:443/....>

# HTML and DOM

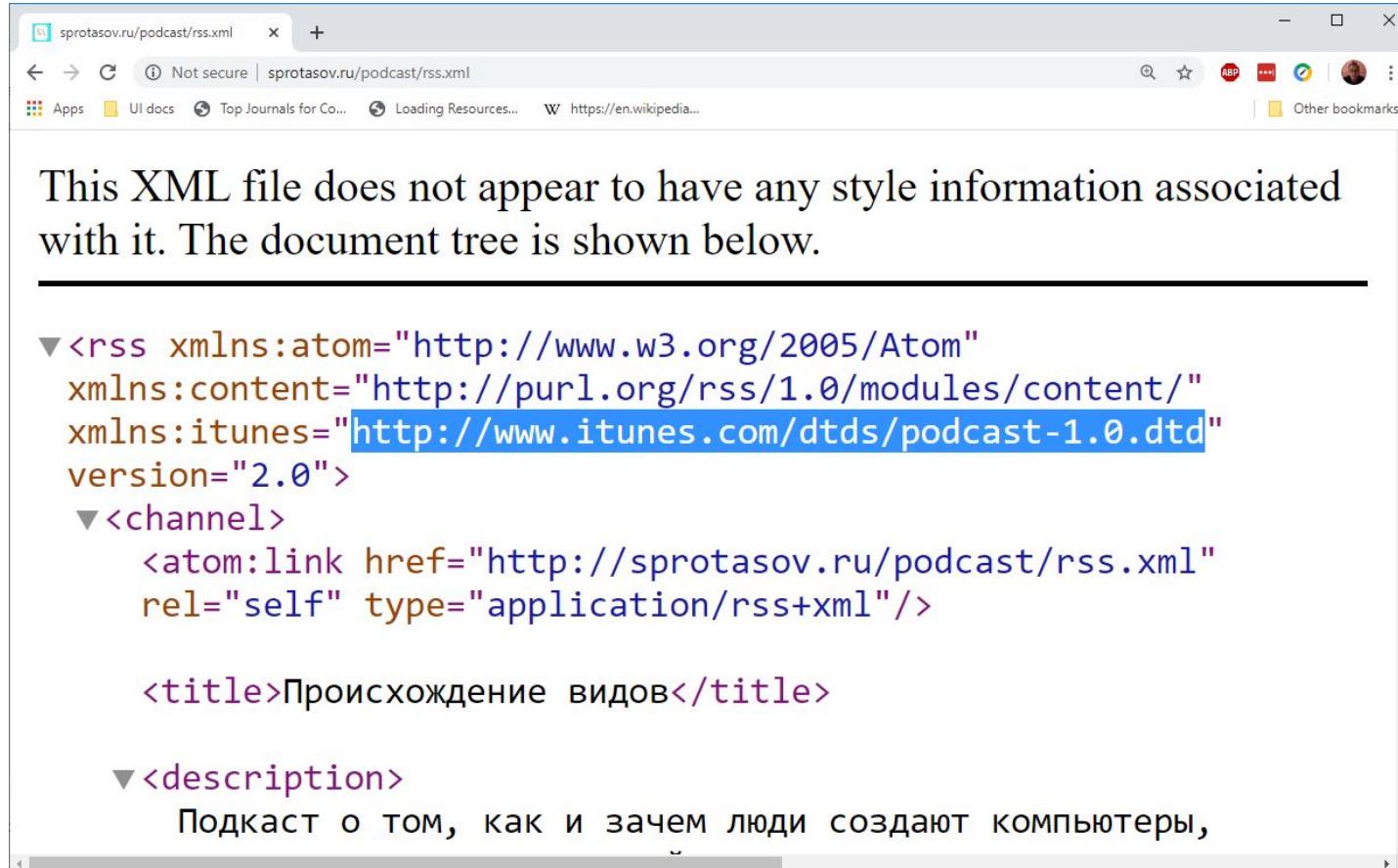
# LANGUAGE TYPOLOGY

- SGML – meta language for description of markup languages. It defines
  - Allowed symbol alphabet (SGML declaration)
  - DTD (data type definition) – markup syntax + semantics
- XML – simplified subset of SGML
  - XML Schema languages (DTD, W3C XSD)
- HTML – application of SGML (initially)
  - xHTML – application of XML

# SCHEMA LANGUAGES

- **DTD, XML Schema** – define document structure and node constraints
- Used for
  - Defining semantic rules (for values, number of children...)
  - Document pre-validation
- Document can be:
  - **type-valid** – meet all DTD constraints
  - **tag-valid** – meet all [SGML/XML] tag constraints

# RSS - Rich Site Summary



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<rss xmlns:atom="http://www.w3.org/2005/Atom"  
      xmlns:content="http://purl.org/rss/1.0/modules/content/"  
      xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"  
      version="2.0">  
  ▼<channel>  
    <atom:link href="http://sprotasov.ru/podcast/rss.xml"  
              rel="self" type="application/rss+xml"/>  
  
    <title>Происхождение видов</title>  
  
    ▼<description>  
      Подкаст о том, как и зачем люди создают компьютеры,
```

# DTD EXAMPLE

```
<!ELEMENT people_list (person*)>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA) >
<!ELEMENT birthdate (#PCDATA) >
<!ELEMENT gender (#PCDATA) >
<!ELEMENT socialsecuritynumber (#PCDATA) >
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
    <name>
      Fred Bloggs
    </name>
    <birthdate>
      27/11/2008
    </birthdate>
    <gender>
      Male
    </gender>
    <socialsecuritynumber>
      1234567890
    </socialsecuritynumber>
  </person>
</people_list>
```

```
<!-- Валидация простого HTML 4.01 -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
```

# XML SCHEMA (XSD) EXAMPLE

```
<?xml version="1.0" encoding="utf-8"?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema">
    <xss:element name="country">
        <xss:complexType>
            <xss:sequence>
                <xss:element name="country_name" type="xss:string"/>
                <xss:element name="population" type="xss:decimal"/>
            </xss:sequence>
        </xss:complexType>
    </xss:element>
</xss:schema>
```

```
<?xml version="1.0" encoding="utf-8"?>
<country>
    <country_name>France</country_name>
    <population>59.7</population>
</country>
```

# HTML5 DOCUMENT STRUCTURE

```
1  <!doctype html>
2  <html>
3      <head>
4          <title>my title</title>
5      </head>
6      <body>
7          body
8              <footer>
9                  <!-- html5 specific -->
10                 footer
11                 </footer>
12             </body>
13         </html>
```

# \*ML-DOCUMENT PARSING METHODS

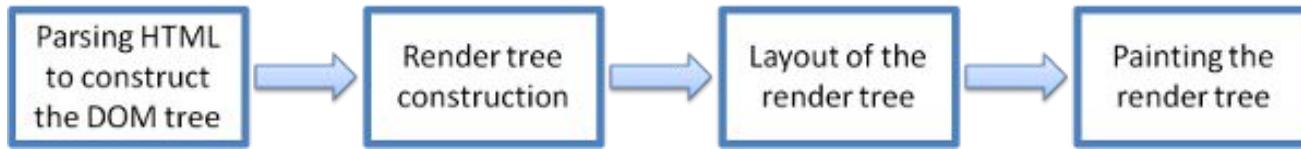
## SAX (Simple API for XML)

- Raises an event/error when new element (token) appears (considering document as stream of tokens and errors)
- Works either as
  - callback-methods (push) or
  - cursor (pull, StAX)
- Requires constant memory
  - Good for embedded systems
- Does not know anything about document's model

## DOM (Document object model, DOM tree)

- Creates full document model
- Used in browsers
- Unpredictable memory usage
  - XML-bombs using DTD
- Query languages (CSS-selectors, xpath, xquery)

# BROWSER ENGINES = LAYOUT ENGINE + JS + ...



- [Good article about browser architecture](#)
- Browser Layout Engine (html + css)
  - **Trident** (IE), "Edge" (Spartan) → Chromium (2019)
  - **Gecko** (Mozilla)
  - **WebKit** (Safari, Chromium-family), WebCore
    - **Blink** (Chrome 28+, Opera 15+, Chrome for Android)
  - Others (KHTML, ~~Presto~~)

# HTML TAGS

- Tag
  - tag name from HTML Schema - mandatory
    - <td> .. </td>
  - Closing tag - mandatory
    - <div> .... </div>
    - 
  - Attributes – define semantics
    - <div id="div1" style="border:1px" class="myDiv">  
    </div>
  - Can have inner tags or inner content (text)
    - <script> console.log(text); </script>
    - <div>  
        <span> ... </span>  
        <input />  
    </div>
  - Layout of the tag is defined by the style
    - In attribute
    - In CSS specification
    - By default

# STYLE

- inline-styles

```
<div  
    style="border: 1px solid gray; color: red"/>
```

- styles inside document
  - <head><style> ..... </style></head>
- styles in separate CSS file

```
○ <link  
    rel="stylesheet" type="text/css"  
    href="xxx.css"/>
```

# STYLE SYNTAX

```
some css selector1
{
    property1: value1 ;
    property2: value2 ;
}

some css selector2
{
    property3: value3 ;
    property4: value4 ;
}
```

Small comment on  
http, html, and privacy

# Browser fingerprint: visited site

# FingerprintJS: 99.5% accuracy [bonus: fresh leak of Safari]

- navigator.userAgent, navigator.language
- new Date().getTimezoneOffset()
- screen.height, screen.width, screen.colorDepth
- HTML5 features support (yes/no)
- doNotTrack/DNT flag (111111), cpuClass, platform
- Installed extensions
- canvas fingerprint (draw on canvas and toDataURL()) — fonts depend on platform
- WebGL fingerprint (for iOS)
- Installed fonts

← → С

https://www.google.com

★ ✉ ≡

Почта Картинки

Войти

Инспектор Консоль Отладчик Сеть Стили ...Поиск URL|| 🔍 ✖  Отключить кэш Без ограничения ⚙️Все HTML CSS JS XHR Шрифты Изображения Медиа WS Прочее

Статус	Мет...	Домен	Файл	Инициатор	Тип	Передано	Раз...
200	GET	🔒 www.goo...	/	document	html	35,08 КБ	107,...
204	POST	🔒 www.goo...	gen_204?atyp=i&ei=g0jlYauoJYORrwSYzc...	m=cdos,cr,d...	html	357 б	0 б

26 запросов | 1,33 МБ / 43,31 КБ передано | Передано за: 39,41 с | DOMContentLoaded: 238 мс | load: 830 мсЗаголовки Куки Запрос Ответ Тайминги ЗащитаПоиск заголовковБлокировать Повторить отправку

Cookie: 1P\_JAR=2022-01-17-10; NID=511=iPBuZr8V\_F5R0F5ve2gCtbXBA5hb0LLrKExdV6WyGSdpaGRq1OWgCLN2ZtTS8K-nyoMEqcQ3wYKuvvvp6N5J-SXiVfc6GvltHCgFf\_rJK4pGzUpdlrKjEW72ezdsK-UAYVxZyhXTXWRCrgrp8sxkh5x9yNEUJY8\_dhj3KdSaKj8; ANID=AHWqTUmz4IB8UqFqx6LSaPbVkJUdnP-wBAHlyjwOXMxH6a7Mj9PiK2PstShMRI64W

DNT: 1

Host: www.google.com  
Sec-Fetch-Dest: document  
Sec-Fetch-Mode: navigate

# Crawling problems

...

## and solutions

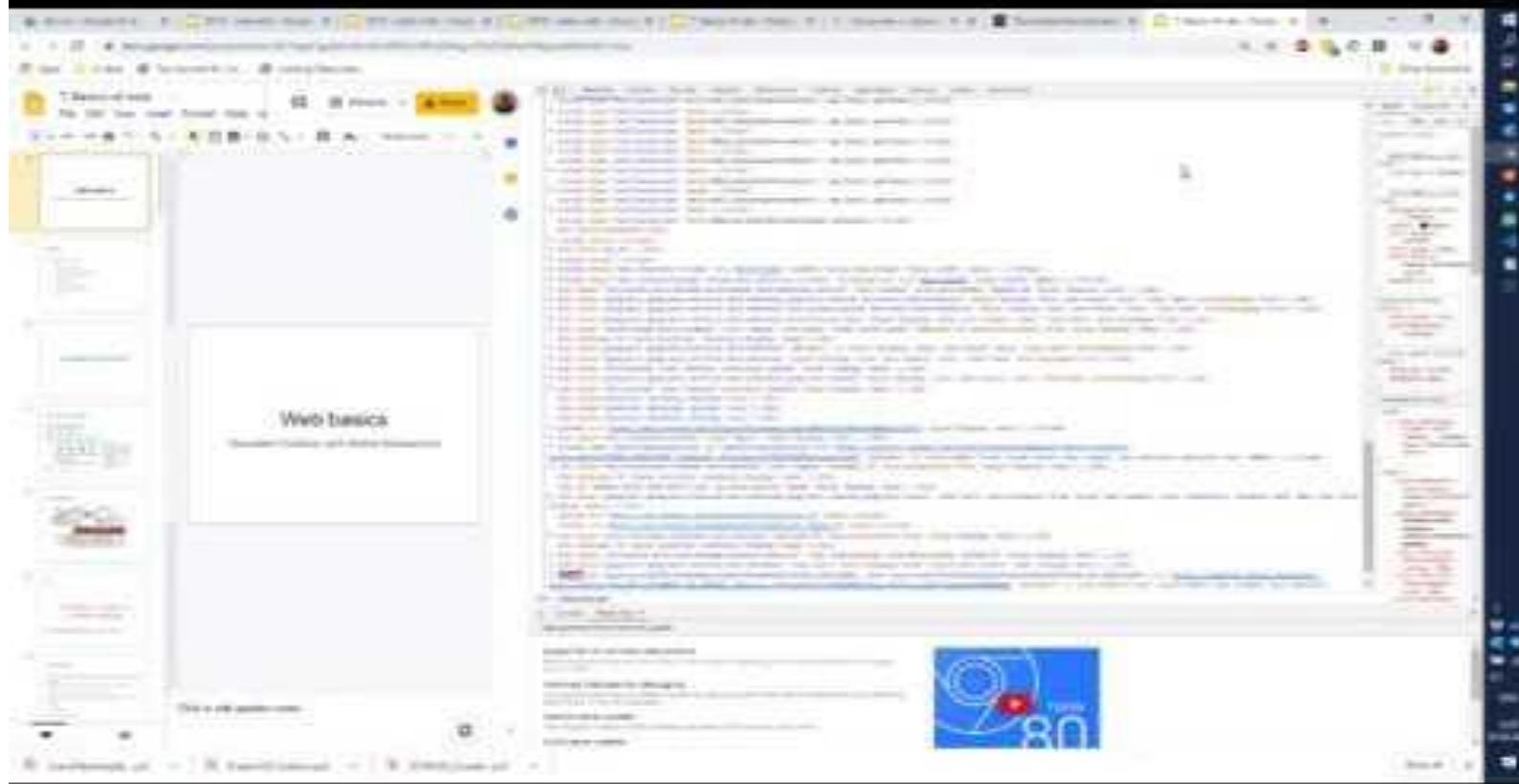
# Problem 1: SaaS vs Documents

On your \*nix machines run:

```
wget --no-check-certificate -O doc.txt http://tiny.cc/00dhkz
```

```
cat doc.txt | sed -e "s/;/;\n/g" | grep "QuadTree search"
```

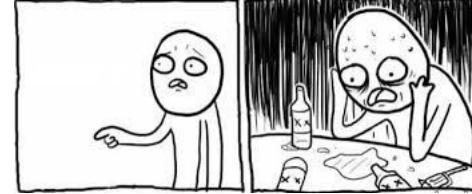
What's wrong?



# How to?

You need a browser engine + JS

1. [Headless] browsers
2. Drivers to manage browsers
3. Automation software:
  - a. [Selenium](#)
  - b. [Puppeteer](#)



# Problem 2: But wait... software engineering?!

How would you parse all links from the Wikipedia article page?

```
import requests  
from bs4 import BeautifulSoup  
  
...
```

**Stop here!** Every big **company knows** that you will parse it's data. It wants to minimize harm you can do. **APIs!** Free anonymous, free authenticated, paid.

1. [Wikipedia API](#)
2. [VK API](#)
3. [Yandex Search API](#)
4. [Google Open Search API](#)
5. ...

# Problem 3. I was downloading ... but it stopped working

1. Company **considers** the data (and service) it has as:
  - a. A property
  - b. A competitive advantage
2. Thus, company **protects** its data from grabbing (and services from proxying):
  - a. With API regulations ([Ya](#), ...) — what is the allowed rate
  - b. Etiquette ([Wiki](#))
  - c. Access keys to control grabbing and proxying speed
  - d. Special legal statements that prohibit grabbing ([ASOS](#))
3. To **enforce** you to obey
  - a. Access key restriction
  - b. IP [range] blocking
  - c. Browser **fingerprint** blocking
4. So, to speed or just enable you crawling ...

# Problem 4: The last but not the least...

## Allow and Disallow

- robots.txt prohibits
  - <http://innopolis.ru/robots.txt>
  - <https://yandex.ru/robots.txt>
  - Wiki :)  

```
# Sorry, wget in its recursive mode is a frequent problem.  
# Please read the man page and use it properly; there is a  
# --wait option you can use to set the delay between hits,  
# for instance.  
#  
User-agent: wget  
Disallow: /
```
- sitemap.xml helps. Start with robots : <https://www.kinopoisk.ru/robots.txt>
  - Sitemap: <https://www.kinopoisk.ru/sitemaps/sitemap.xml>

# Crawl safe!



# Quality assessment

Stanislav Protasov

# Agenda

## Metrics:

- Offline
- Online
- Other aspects

**Q: what is the target metric of the service?**

# What is the target metric of a service?

- Money
  - Traffic share (yandex vs google)
  - User satisfaction
  - User happiness
  - Logins/Subscriptions
  - ...
- 1) Unfortunately, we cannot replay new ML models with complex human target behavior (e.g. *if we had this feature, we would attract \$\$*)
  - 2) Some targets are hard to compute online (delays, is it measurable?, ...)

## What we can?

Create a set of easily **computable** metrics which correlate with target ones.

*Satisfaction (?) vs probability to find relevant doc*

*Traffic Share (delayed) vs MAU*



# High-level evaluation techniques

Offline evaluation — result of a new model is compared with manually [pre]processed data

- **On a bucket:** accuracy, precision, recall
- **By assessors:** relevance scale
  - (assessors are humans: kappa stats, weighted, majority voting, ...)

**Online evaluation** — a new model is *compared* with an old one by some *target metric* on a subset of users. Most widely used approach is A/B testing

BTW, what is *relevance*?

**Information need** is encoded in a **query**.

Query is used to get **results**.

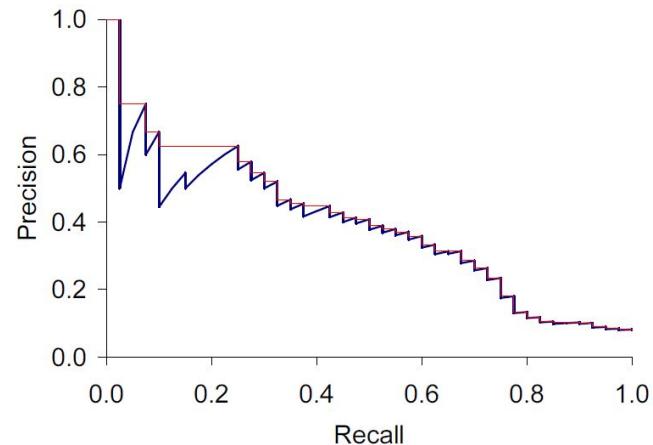
But do results **satisfy information need**?

{0, 1} or [0..1]: {IR, REL-, REL+}

# Offline: accuracy, precision, recall, $F_1$

- **Accuracy** is not the case
- **Precision** is how many relevant out of retrieved
- **Recall** is how many relevant retrieved out of all relevant\*
- **Precision-recall curve** can be used for ranked results
- $F_1$  is to come up with a single number

Users are *tolerant* to irrelevant results



# Mean Average Precision

Precision@K == **TruePositive@K** / K

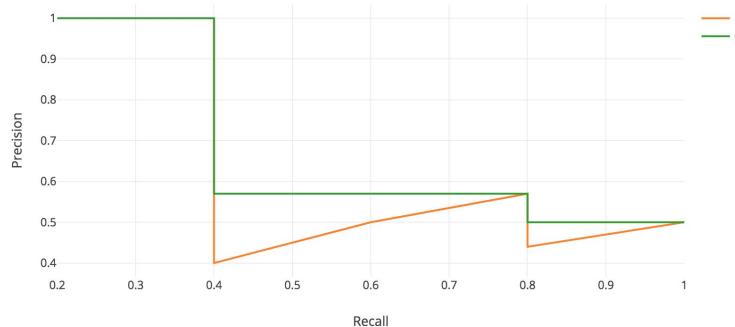
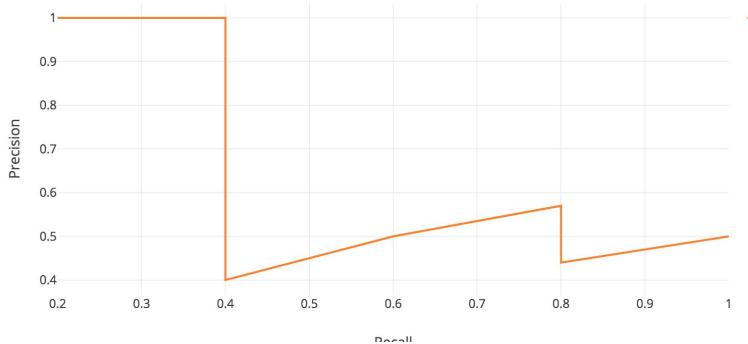
Recall@N\* == **TP@N** / ~~AllPositive~~ **TP@K**

Average Precision: AP =  $\int_0^1 p(r)dr$

1. Improve “steps”

2. AP@n      AveP =  $\sum_{k=1}^n P(k)\Delta r(k)$

$$MAP = \frac{\sum_{q=1}^Q AveP(q)}{Q}$$



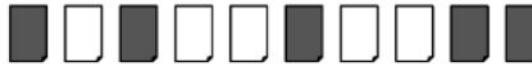
# MAP

---



= relevant documents for query 1

Ranking #1



Recall 0.2 0.2 0.4 0.4 0.4 0.6 0.6 0.6 0.8 1.0

Precision 1.0 0.5 0.67 0.5 0.4 0.5 0.43 0.38 0.44 0.5



= relevant documents for query 2

Ranking #2



Recall 0.0 0.33 0.33 0.33 0.67 0.67 1.0 1.0 1.0 1.0

Precision 0.0 0.5 0.33 0.25 0.4 0.33 0.43 0.38 0.33 0.3

$$\text{average precision query 1} = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$

$$\text{average precision query 2} = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$\text{mean average precision} = (0.62 + 0.44)/2 = 0.53$$

$$\text{AveP} = \sum_{k=1}^n P(k)\Delta r(k)$$

# Whiteboard time (MAP)

Q	<i>“vk”</i>	<i>“Good search engine”</i>	<i>“Fanny Animal Images”</i>	<i>“Who let the dogs out?”</i>
1	<b>vk.com</b>	Good engine repair	Images of Fanny Ardent	<b>Baha Men - Who Let The Dogs Out (Youtube)</b>
2	<b>vkusvill</b>	<i>yahoo.com</i>	<b>9GAGs</b>	CNN: old men let the dogs out to prevent robbery
3	<b>МЛ</b>	<i>google.com</i>	<b>fishki.net</b>	<i>Who Let The Dogs Out Wikipedia</i>
4	Some trash	<b>yandex.ru</b>	CNN.com	Funny Dogs website
5	<i>facebook.com</i>	<i>altavista</i>	<i>Moscow Zoo</i>	<b>Baha Men - Who Let The Dogs Out (Lyrics)</b>

## Offline: simple is ok

*Mean reciprocal rank* (MRR):  $\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$

$Q$  — bucket of queries

$\text{rank}_i$  — rank of the **first** relevant document

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$

# Whiteboard time (MRR)

Q	<i>“vk”</i>	<i>“Good search engine”</i>	<i>“Fanny Animal Images”</i>	<i>“Who let the dogs out?”</i>
1	<b>vk.com</b>	Good engine repair	Images of Fanny Ardent	<b>Baha Men - Who Let The Dogs Out (Youtube)</b>
2	<b>vkusvill</b>	<i>yahoo.com</i>	<b>9GAGs</b>	CNN: old men let the dogs out to prevent robbery
3	МЛ	<b>google.com</b>	<b>fishki.net</b>	<i>Who Let The Dogs Out Wikipedia</i>
4	Some trash	<b>yandex.ru</b>	CNN.com	Funny Dogs website
5	<b>facebook.com</b>	<i>altavista</i>	<i>Moscow Zoo</i>	<b>Baha Men - Who Let The Dogs Out (Lyrics)</b>

# Offline: discounted gain model

**Cumulative gain CG@p:**  $CG_p = \sum_{i=1}^p rel_i$  ( $p$  — e.g. 10 items on SERP,  $rel_i$  - 0/1 or 0..1)

**Discounted CG@p:**  $DCG_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^p \frac{rel_i}{\log_2(i+1)}$        $DCG_p = \sum_{i=1}^p \frac{2^{rel_i} - 1}{\log_2(i+1)}$

**Normalized DCG:** divide DCG by the best possible achievable (Ideal) DCG:

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

$$DCG_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^p \frac{rel_i}{\log_2(i+1)}$$

# DCG example

	Relevance	Discounted Gain
1	R	1
2	R	1
3	N	0
4	N	0
5	R	$\frac{1}{\log_2(r+1)}$
6	R	0,38
7	N	0,35
8	R	0
9	N	0,31
10	N	0
...	...	0

$$DCG_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^p \frac{rel_i}{\log_2(i+1)}$$

# \* Whiteboard time (DCG)

Q	<b>“vk”</b>	<b>“Good search engine”</b>	<b>“Fanny Animal Imajes”</b>	<b>“Who let the dogs out?”</b>
1	<b>vk.com</b>	Good engine repair	Images of Fanny Ardent	<b>Baha Men - Who Let The Dogs Out (Youtube)</b>
2	<b>vkusvill</b>	<b>yahoo.com</b>	<b>9GAGs</b>	CNN: old men let the dogs out to prevent robbery
3	<b>МЛ</b>	<b>google.com</b>	<b>fishki.net</b>	<i>Who Let The Dogs Out Wikipedia</i>
4	Some trash	<b>yandex.ru</b>	CNN.com	Funny Dogs website
5	<b>facebook.com</b>	<b>altavista</b>	<b>Moscow Zoo</b>	<b>Baha Men - Who Let The Dogs Out (Lyrics)</b>

## Offline: pFound

**pFound** idea is similar. Estimate probability to find relevant item.

Presets:

- **max relevance is 0.4** (R), otherwise is 0 (NR, maybe R);
- probability to quit with no reason **pBreak** - 0.15;
- User watches from top to bottom, stops if *found* relevant OR if *tired*;

$$pLook[i] = pLook[i-1] * (1 - pRel[i-1]) * (1 - pBreak)$$

$$pfound = \sum_{i=1}^n pLook[i] * pRel[i]$$

$$pLook[i] = pLook[i-1] * (1 - pRel[i-1]) * (1 - pBreak)$$

$$pfound = \sum_{i=1}^n pLook[i] * pRel[i]$$

Q	<i>"Fanny Animal Images"</i>
1	Images of Fanny Ardent
2	<b>9GAGs</b>
3	<b>fishki.net</b>
4	CNN.com
5	<i>Moscow Zoo</i>

$$pLook[1] = 1$$

$$pLook[2] =$$

$$pLook[3] =$$

$$pLook[4] =$$

$$pLook[5] =$$

$$pFound =$$

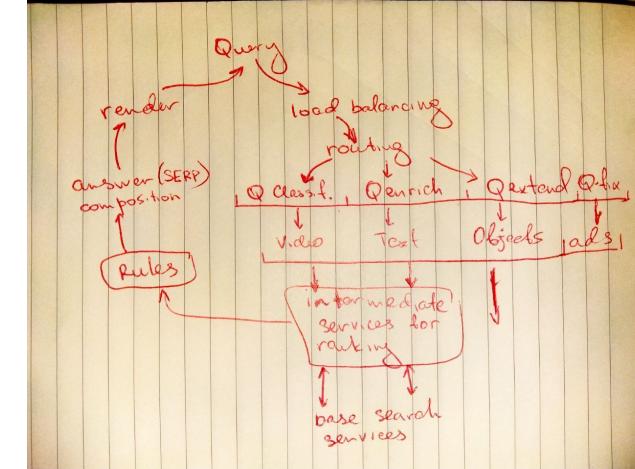
# Online: A/B testing

Online testing allows to test of a *target metric*!

1. Is *relative*. Create **alternative** model
2. Formulate **target metric** (CTR, dwell-time, total revenue, ARPU)
3. Prepare **representative subset of users** (for complex setups use multinomial experiments)
4. Run controlled experiment, **aggregate statistics** on your metric
5. Run **statistical test** (Welch, Fisher, Student, ...) to see that **mean shift** for a distribution is significant\*
6. Make a **decision** to accept (no always “better” for new features)

# A/B testing infrastructure

- User **identification** method (logged in, cookies, fingerprints)
- **Experiment support** in the whole infrastructure (should not be as usual traffic) with flags/parameters
- Mapping **registry** (users to experiments)
- **Statistical tools** ready before you start experiment



**Use a platform:** Google Analytics, Optimizely, VWO, ...

## Other quality aspects

New users should get most ranked items, whereas old users should be **surprised** with quality items from “long tail” [[ref](#)]

Add **diversity** to recommendations, get out of the bubble [[ref](#)]

**Novelty** issue (did I see this before?) [[ref](#)]

Stay **legally** and **ethically** safe (no medical questions, no porn, no swearing)

# ~~Search engine~~ One company's target

User should **solve a problem** with a service: get an answer to the question, a service or an item without leaving portal. Steps to achieve:

- Keep users *logged in*
- Evaluate user intent (surfing, buying, asking, ...)
- Provide a *quality service* for each intent (first, specific search, then specific service)
- Don't be evil

**DON'T BE  
EVIL\***

**\*Unless It's Profitable**

**Google**

# Building an inverted index

Stanislav Protasov

# Agenda

1. Languages
  - a. Formal approach
  - b. Tokenization
  - c. Stemming and lemmatization
2. Building an inverted index

# Languages

# Languages: syntax, semantics, pragmatics

- Pragmatics: new\_var = map(lambda x: x - 2, [4, 5, 6])
- **Semantics:**
  - This is a valid sentence in English.
  - The worst part and clumsy looking for whoever heard light.
  - Twas brillig, and the slithy toves did gyre and gimble in the wabe.
  - Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor ...
- **Grammar (syntax):**
  - I can has cheezburger?
  - I nevr mkae tipos and erors in my sentencs.
  - I'm chuffed to bits seeing you! Do ya wanna watch some telly together, bro?
  - I'll txt w/my ETA 2U.

# Questions to discuss?

Where does semantics hide in the language?

What is the purpose to have a language syntax?

# Syntax

# Definitions

**syntax** guards word order — initially linguistic term

[formal] **grammar** - describes *how to* form strings from a language's *alphabet* that are valid according to the language's *syntax*. = set of **rules**, way to express syntax

**formal language** - set of all strings *allowed* by a grammar. = **satisfy** rules

Grammar is a  $\langle \Sigma$  - terminals, **N** - nonterminals, **P** - productions, **S** - start symbol $\rangle$

ABC – nonterminals  
 abc – terminals  
 $\alpha\beta$  – seq of terminals/non-terminals  
 $\gamma$  – non-empty seq

## Chomsky Normal Form (CNF) and Chomsky hierarchy

0. Recursively enumerable (almost any productions)

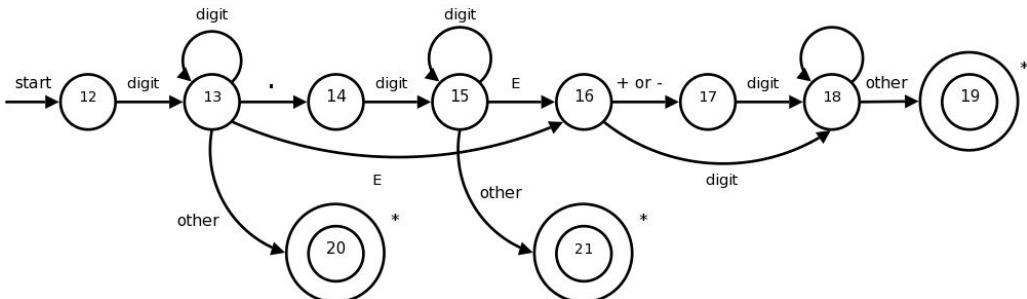
1. Context-sensitive  $\alpha A \beta \rightarrow \alpha \gamma \beta$

Also noncontracting grammar ( $\alpha \rightarrow \beta$ , where  $\alpha, \beta \in (\Sigma \cup N)^*$  and  $|\alpha| \leq |\beta|$ )

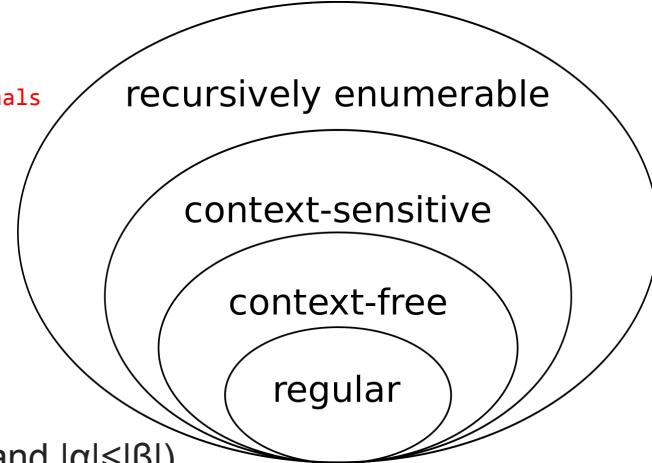
E.g. **Professor** ( $\alpha$ ) **Brown** (proper noun) vs **Brown** (adjective) **bear** ( $\beta$ )

2. Context-free  $A \rightarrow \alpha$

3. Regular  $A \rightarrow a$  or  $A \rightarrow aB \mid A \rightarrow Ba$  (exceptionally)



— e.g. numbers.



# Extended Backus-Naur Form (EBNF)

A = B,C. # concat

A = B|C|D. # one of

A = [B]. # 0/1

A = {B}. # 0+

A = B{B}. # 1+

A = (B|C)(D|E). # grouping (to avoid service NT)

# Syntax tree

D = the | a

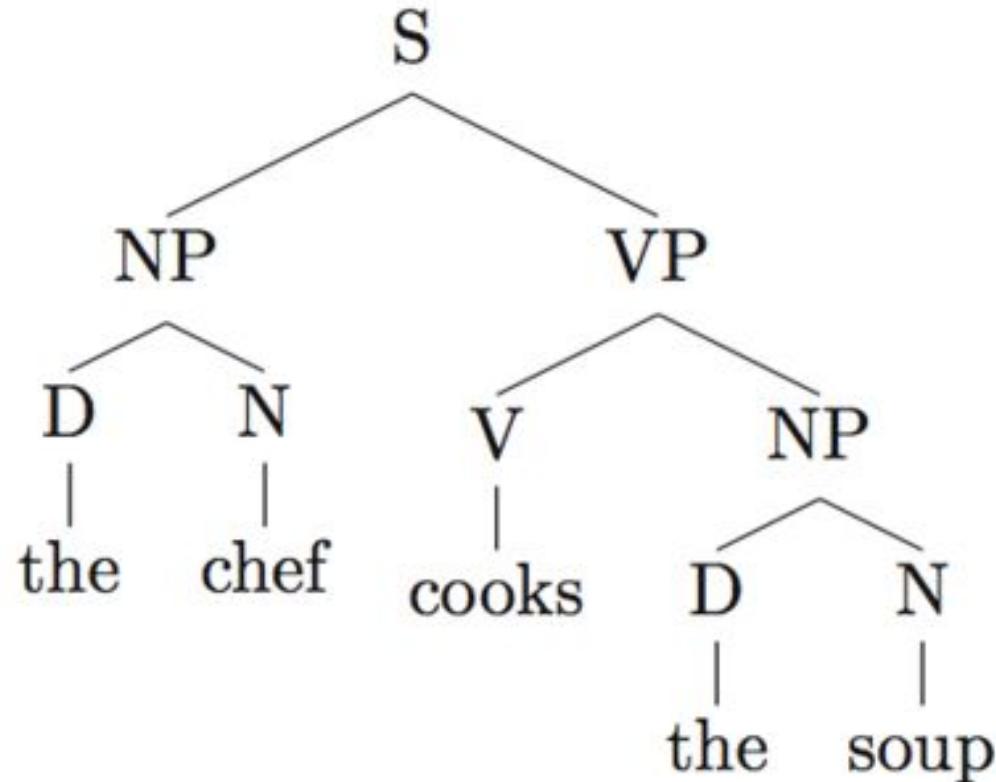
N = chef | soup

V = cooks

NP = D N | ...

VP = V N | V NP

S = NP VP



< $\Sigma$  - terminals, **N** - nonterminals, **P** - productions, **S** - start symbol>

# Tokenization

**Lexemes** (distinct objects of the language) are produced by **scanner**.

**token = (lexeme, token\_type ~ PoS)**

Program, converting stream of characters into a stream of **tokens** is called **lexical analyzer, lexer, tokenizer**.

```
i like to read science fiction.  
[('i', 'PRP'), ('like', 'VBP'), ('to', 'TO'),  
 ('read', 'VB'), ('science', 'NN'), ('fiction'  
, 'NN'), ('.', '.')]
```

# Syntax analysis helps proper tokenization

*L'ensemble* □ one token or two? *L* ? *L'* ? *Le* ?

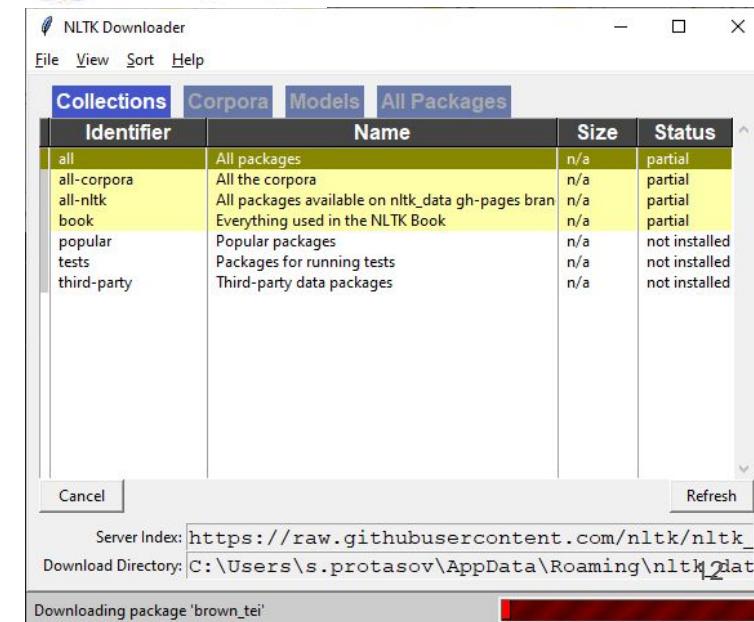
莎拉波娃现在居住在美国东南部的佛罗里达

اسْتَقْلَتُ الْجَزَائِرُ فِي سَنَةِ 1962 بَعْدَ 132 عَامًا مِنَ الْاحْتِلَالِ الْفَرْنَسِيِّ.

```
# Language-specific punctuation
import nltk
nltk.download()
st = nltk.data.load('tokenizers/punkt/english.pickle')
st.tokenize(text)      # sentence splitting

nltk.download('punkt')
nltk.word_tokenize()  # language specific
```

```
# grammar based tokenization
simple_grammar = nltk.parse_cfg(...)
parser = nltk.ChartParser(simple_grammar)
trees = parser.nbest_parse("A car has a door")
```



# ... or use statistics from corpora!

This is a sentence that we will use to test the magic tool

```
nltk.UnigramTagger(brown_tagged_sents).tag(tokens)
```

```
[('This', 'DET'), ('is', 'VERB'), ('a', 'DET'), ('sentence', 'NOUN'), ('that', 'ADP'), ('we', 'PRON'), ('will', 'VERB'), ('use', 'NOUN'), ('to', 'PRT'), ('test', 'NOUN'), ('the', 'DET'), ('magic', 'ADJ'), ('tool', 'NOUN')]
```

```
nltk.BigramTagger(brown_tagged_sents).tag(tokens)
```

```
[('This', 'DET'), ('is', 'VERB'), ('a', 'DET'), ('sentence', 'NOUN'), ('that', 'PRON'), ('we', 'PRON'), ('will', 'VERB'), ('use', 'VERB'), ('to', 'PRT'), ('test', 'VERB'), ('the', 'DET'), ('magic', 'ADJ'), ('tool', 'NOUN')]
```

We are almost ready to build an index of tokens. Anything left?

# Compression techniques used across methods

- **case folding:** London = london; Лев = лев
- **stemmer vs lemmert:**
  - stemming:* compress = compression = uncompressed  
бегу = бег
  - lemmatization:* better = good  
бегу = бегать
- ignore **stop words:** to, the, it, be, or, ...
  - Problems arise when search on “To be or not to be” or “the month of May”
- **Thesaurus:** fast = rapid; лев = лёвушка
  - handbuilt clustering

# Stop here!

1. Language, Syntax, Grammar
2. Formal languages, grammars and types
3. Tokenization and syntax analysis
4. Stems and stemming

# Boolean retrieval

BIR is based on **Boolean logic** and classical **set theory** in that both the documents to be searched and the user's query are conceived as sets of terms (a **bag-of-words model**). Retrieval is based on **whether or not the documents contain the query terms.**

(Wikipedia), [[The Book](#)]

- Boolean query
  - E.g., (“obama” AND “healthcare” AND NOT “news”)

# Search and recommender systems idea

**Boolean retrieval**, exact nearest neighbour search or exact range queries can be too **expensive**. *Can we do faster?*

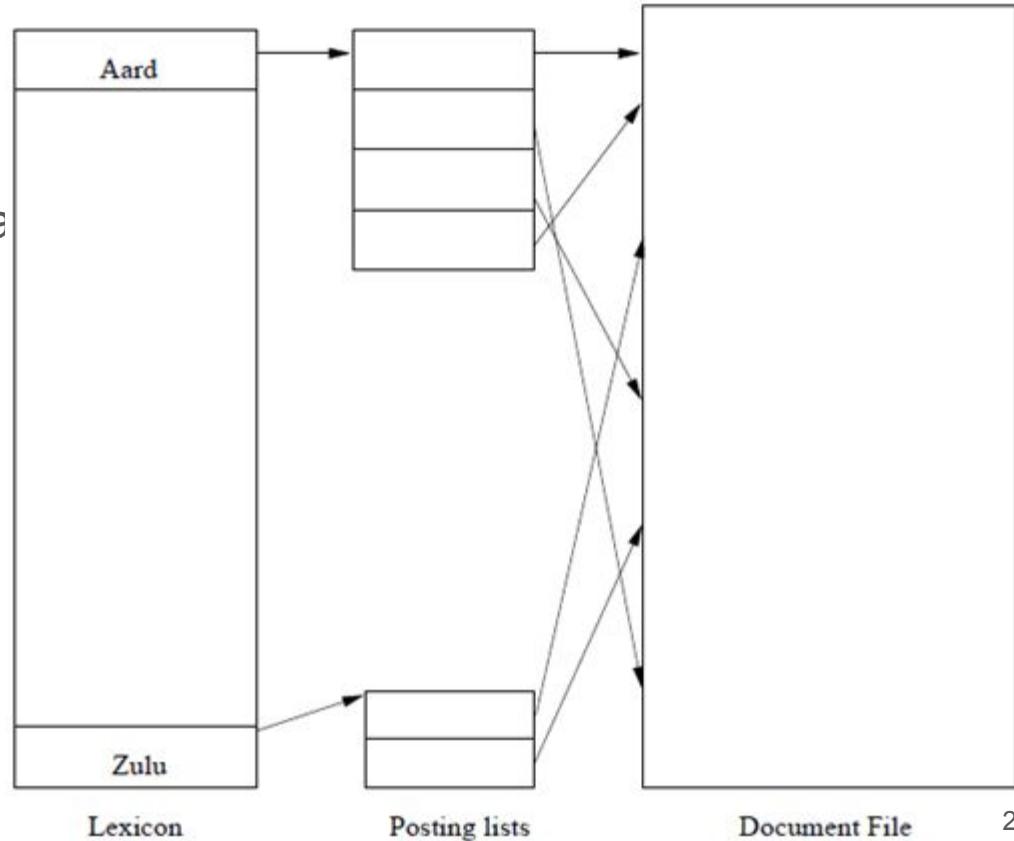
**Pre-select** (*pre-ranking sets*, approximate NN ...) which is done fast (e.g.  $O(\log(N))$ ), selects enough to catch [almost] all relevant elements. Requires data structures: ***indices***.

**Select** (*ranking*, exact match) is done on a smaller set (PRS).

# Text indexing: inverted index

# Inverted index

- 1) Build a **lexicon** for the whole database
- 2) For each word of lexicon build a **posting list** (set of pointers)
- 3) [optional] persist this structure as a sparse matrix

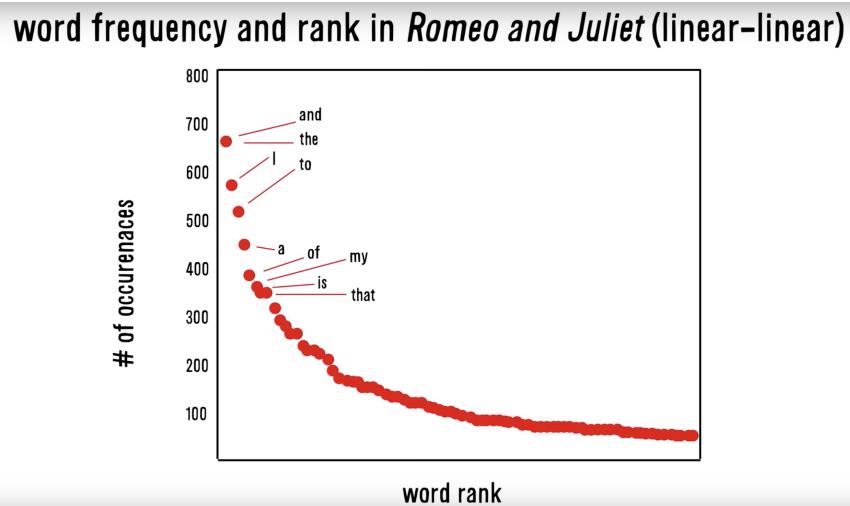


# Remarks on lexicon

- Languages have lots of names and phrasal forms of 2+ words (e.g. New York) → use **bigrams** and alternative tokenizations (e.g. subword tokenization)
- Stopwords should be selected carefully

# Document frequency

Idea: a term is more discriminative if it occurs only in fewer documents



<https://medium.com/datadriveninvestor/zipfs-law-breakdown-application-in-app-development-5e9cda70cdc8>

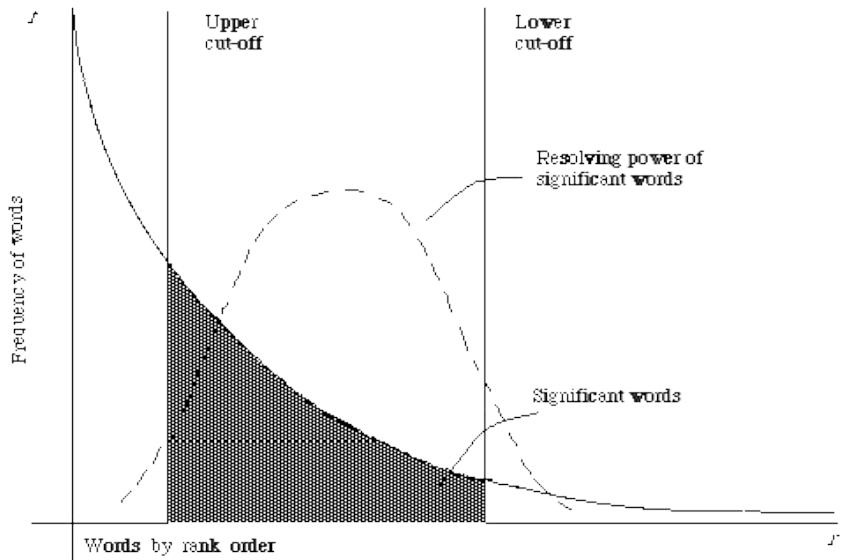


Figure 2.1. A plot of the hyperbolic curve relating  $f_i$ , the frequency of occurrence and  $r_i$ , the rank order. (Adapted from Schulte<sup>4</sup> page 120)

# Subjective remarks on dictionary construction

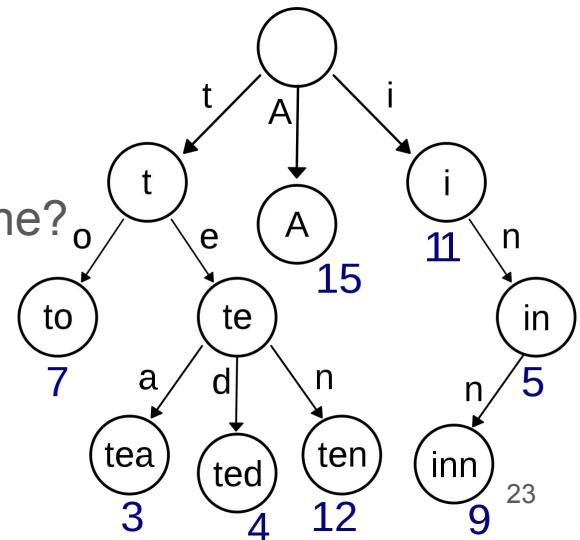
External memory dictionaries are almost useless\* for real-time applications. Files are used for persistence (mmap)

Index **building** requires more memory, than index storage itself. Map-Reduce is widely used for this. (Book, 4.4)

Indices are usually **static**. For dynamic read (Book, 4.5)

What if total number of documents is big for a single machine?

- Use Trie
- Use Sharding: *hash based, lexicographical*



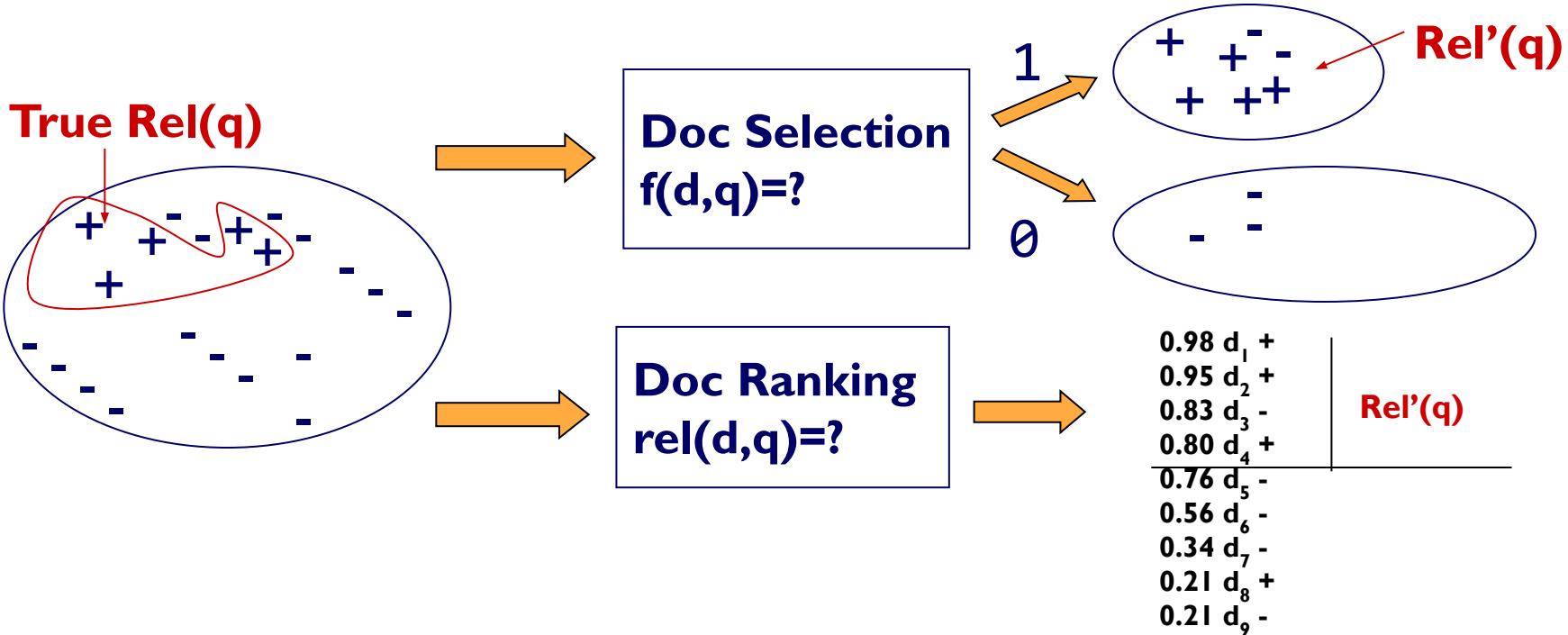
# Search with Boolean query

- **Boolean query**
  - E.g., “obama” AND “healthcare” AND NOT “news”
- **Pre-ranking set**
  - Lookup query term in the dictionary
  - Retrieve the posting lists
  - Operation
    - AND: intersect the posting lists ([skip-lists can help to intersect in O\(m+n\)](#))
    - OR: union the posting list
    - NOT: diff the posting list
- **“Ranking”: Last step**
  - Re-check selected documents hold **expected substring** (for query search)

# Deficiency of Boolean model

- The query is unlikely precise
  - “**Over-constrained**” query (terms are too specific): no relevant documents can be found
  - “**Under-constrained**” query (terms are too general): over delivery
  - It is hard to find the right position between these two extremes (hard for users to specify constraints)
- Even if it is accurate
  - Not all users would like to use such queries
  - All relevant documents are **not equally** important
    - No one would go through all the matched results
- Relevance is a matter of degree!

# Document Selection vs. Ranking



# Ranking is often preferred

- Relevance is a matter of degree
  - Easier for users to **find appropriate queries**
- A user can stop browsing anywhere, so the **boundary is controlled by the user**
  - Users prefer *coverage* would view more items
  - Users prefer *precision* would view only a few
- Theoretical justification: Probability Ranking Principle
  - relevance has a probabilistic interpretation. According to this principle documents are ranked by a probability  $p(\text{Rel}|d, q)$ , where  $\text{Rel}$  denotes the event of a document  $d$  being relevant to a query  $q$

# Retrieval procedure in modern IR

- Boolean model provides all the ranking candidates
  - Locate documents satisfying (somehow) Boolean condition
    - E.g., “obama healthcare” -> “obama” **OR** “healthcare”
- Rank candidates by **relevance**
  - Important: the definition of relevance
- Efficiency consideration
  - Top-k retrieval ([Google](#) example of page 110)

# Intuitive understanding of relevance

	information	retrieval	retrieved	is	helpful	for	you	everyone
Doc1	1	1	0	1	1	1	0	1
Doc2	1	0	1	1	1	1	1	0
Query	1	1	0	0	0	0	0	0



*E.g., 0/1 for Boolean models,  
probabilities for probabilistic models*

# Ranking over Inverted Index: TF-IDF

Term frequency  $\text{tf}(t,d)$ , count of a term  $t$  in a document  $d$ .

- Boolean "frequencies":  $\text{tf}(t,d) = 1$  if  $t$  occurs in  $d$  and 0 otherwise;
- **term frequency adjusted** for document length :  $f_{t,d} \div (\text{number of words in } d)$
- **logarithmically scaled** frequency:  $\text{tf}(t,d) = \log(1 + f_{t,d})$
- augmented frequency

$$\text{tf}(t, d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

Inverse document frequency  $\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

# Home reading

The book, chapter 4 (index construction)

# Spell-checking, query correction, wildcards

Stanislav Protasov

# Agenda

- Wildcards and regexp support
  - Wildcard types
  - Permuterm
  - Regexp support
- Spell-checking
  - Isolated words
  - Context-dependent approach
  - Soundex

# Wildcards

# Why do we need wildcards?

- uncertain of the spelling of a query term
- aware of multiple variants of spelling (colou?r)
- unsure about part of speech and stemming
- foreign words and spelling (**University** street  
vs **Universitetskaya ulitsa**)

# Syntax of wildcards

- \*? - wildcards with joker symbols
  - *Trailing* wildcard query (**Mos\***) — handled with **search trees**
  - *Leading* wildcard queries (**\*sity**) — handled with **reversed search trees (ytis...)**
  - Can we handle **Mos\*ow** query? **M\*S\*K?**
- SQL
  - column LIKE “%ab\_d”
- Regexp
  - [a-zA-Z][a-zA-Z0-9]{0-30}

# How they do it in databases

“The SQL `LIKE` operator very often causes unexpected performance”

**Only the part before the first wild card** serves as an access predicate. The remaining characters do not narrow the scanned index range—non-matching entries are just left out of the result.

PostgreSQL: The optimizer can also use a **B-tree index** for queries involving the pattern matching operators `LIKE` and `~` if the pattern is a constant and is anchored to the beginning of the string — for example, `col LIKE 'foo%`' or `col ~ '^foo'`, but not ~~`col LIKE '%bar'`~~

LIKE 'WI%ND'	LIKE 'WIN%D'	LIKE 'WINA%'
WIAW	WIAW	WIAW
WIBLQQNPUA	WIBLQQNPUA	WIBLQQNPUA
WIBYHSNZ	WIBYHSNZ	WIBYHSNZ
WIFMDWUQMB	WIFMDWUQMB	WIFMDWUQMB
WIGLZX	WIGLZX	WIGLZX
WIH	WIH	WIH
WIHTFVZNLC	WIHTFVZNLC	WIHTFVZNLC
WIJYAXPP	WIJYAXPP	WIJYAXPP
<b>WINAND</b>	<b>WINAND</b>	<b>WINAND</b>
WINBKYDSKW	WINBKYDSKW	WINBKYDSKW
WIPOJ	WIPOJ	WIPOJ
WICRDK	WICRDK	WICRDK

## Permuterm index

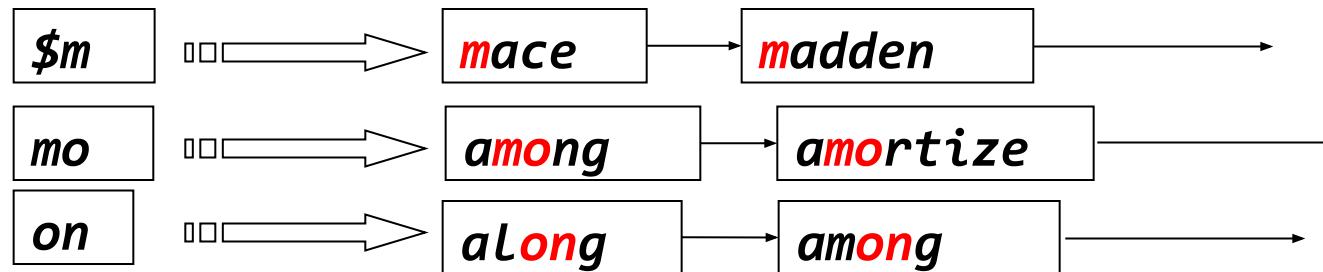
1. Add \$ to the end of the word: hello → hello\$
2. Compose all rotations of this word
  - a. hello\$, ello\$h, llo\$he, lo\$hel, o\$hell, \$hello
3. Build a search tree (B-tree, trie) on this extended vocabulary
4. For a particular query run the following search algorithm (star \* means *exhaustive search* of a subtree, with filtering maybe):
  - a. X lookup on X\$
  - b. X\* lookup on \$X\*
  - c. \*X lookup on X\$\*
  - d. \*X\* lookup on X\*
  - e. X\*Y lookup on Y\$X\* hel\*o?
  - f. X\*Y\*Z – ?

# What's wrong with permuterm?

- Not handling **multiple** joker symbols
- **4-10-times increases** vocabulary size

## K-gram index (q-grams)

1. Mark word start/end with \$
2. 3-grams of `castle` are: \$ca, cas, ast, stl, tle, le\$.
3. Maintain a second inverted index from k-grams to dictionary terms that match each k-gram (here k=2)



4. Convert wildcards into boolean queries

re\*ve → \$re & ve\$.

5. Comment: as words, not all k-grams of fixed k are equally useful (zzz vs the). 9

# Multigrams (V-grams)

Each k-gram can have specific  $k$ : “`<a href`” and “`.mp3`”

**Selectivity of x-gram** is the fraction of data units which contain at least one occurrence of the gram. **Filter factor**.

$$FF = 1 - \text{selectivity}(x)$$

Take only grams with high FF.

Order query parts by filter factor.

*Comment: hard to maintain **online***

## **Algorithm 3.1 Multigram index**

**Input:** database

**Output:** index: multigram index

### **Procedure**

- [1]  $k = 1$ ,  $\text{expand} = \{\cdot\}$  //  $\cdot$  is a zero-length string
- [2] While ( $\text{expand}$  is not empty)
- [3]   k-grams := all  $k$ -grams in database  
              whose  $(k-1)$ -prefix  $\in \text{expand}$
- [4]    $\text{expand} := \{\}$
- [5]   For each gram  $x$  in k-grams
- [6]     If  $\text{sel}(x) \leq c$  Then // check selectivity
- [7]        insert( $x$ , index) // the gram is useful
- [8]     Else
- [9]         $\text{expand} := \text{expand} \cup \{x\}$
- [10]       $k := k + 1$

# Regexp support

## Index support for regular expression search

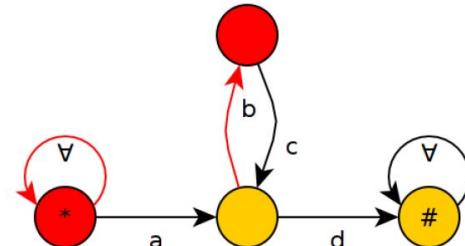
expressing same class of “languages” as finite automata.

General idea is similar:

$/a(bc)^*d/$

xyzabc**c**dxyz

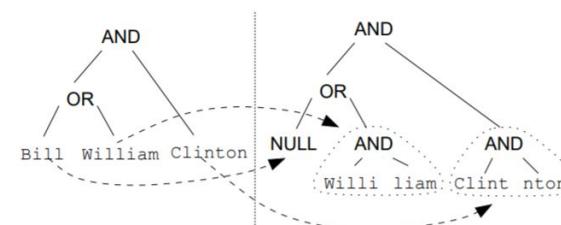
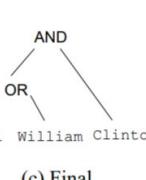
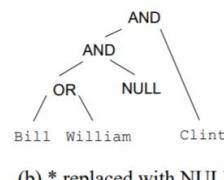
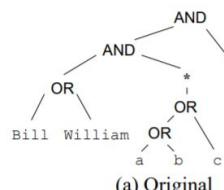
$/[ab]cde/ \Rightarrow (acd \text{ OR } bcd) \text{ AND } cde$



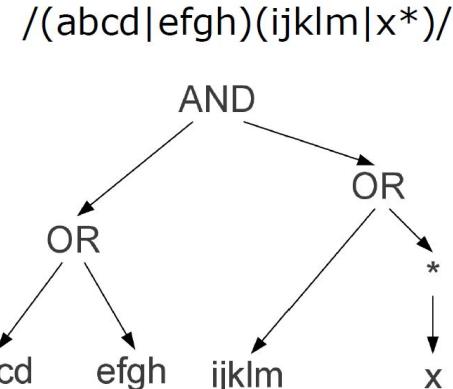
# Regexp support methods: FREE

FREE (2002):

1. Extract tree of continuous string fraction from regex.
  - o  $*$  = NULL, NULL “eats” parent OR
  - o AND eats child NULL
2. Transform those continuous fractions to **multigrams**
3. Use inverted index on multigrams for query evaluation



(b) Final physical access plan



# Regexp support methods: GCS

Regular Expression Matching with a Trigram Index (Google Code Search, 2006)

- Get 5 characteristics about each part of regex: *emptyable, exact, prefix, suffix, match*.
- Recursively union them (with possible simplification)
- Use inverted index of trigrams for query evaluation (similar to pg\_trgm)

Original regex: /a(bc)+d/

a: {exact: a}

bc: {exact: bc}

d: {exact: d}

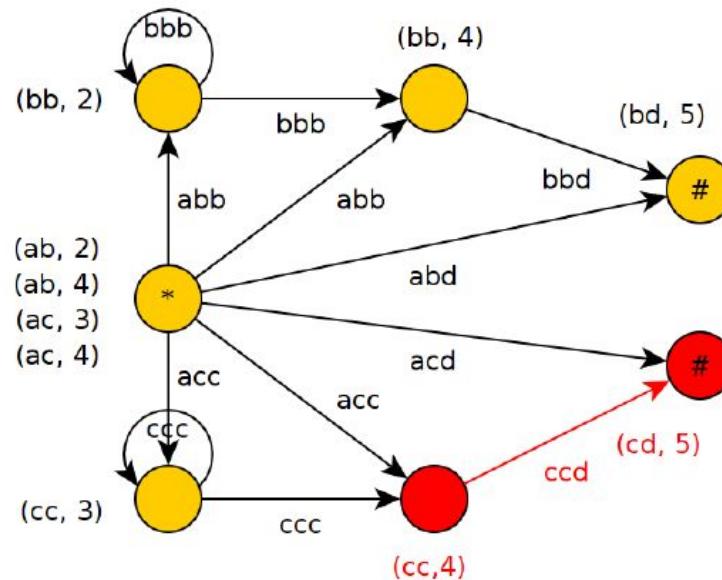
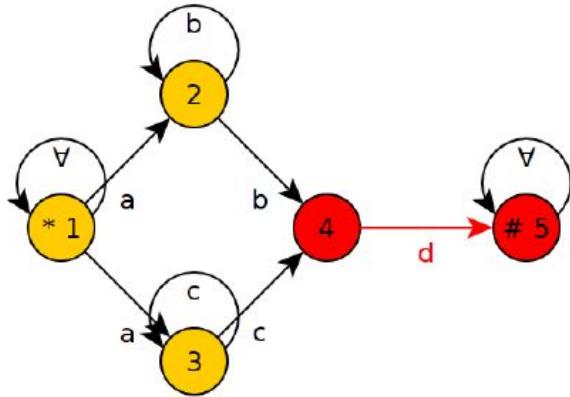
(bc)+: {prefix:bc, suffix: bc}

a(bc)+: {prefix:abc, suffix:bc}

a(bc)+d: {prefix:abc, suffix:bcd} == abc AND bcd

# Regexp support methods: automaton transformation (2012)

1. Procedure for automaton transformation into corresponding graph
2. Procedure to simplify a graph
3. Collect path matrix and convert to binary query



# Query correction

# Spell correction

- Two principal uses
  - Correcting **document(s)** being indexed
  - Correcting user **queries** to retrieve “right” answers
- Two principles:
  - proximity-base (take closest)
  - probability-base (take most frequent)

# Spell correction

- Two main flavors:
  - Isolated word
    - Check each word on its own for misspelling
    - Will not catch typos resulting in correctly spelled words  
e.g., *from* → *form*
  - Context-sensitive
    - Look at surrounding words,
    - e.g., *I flew form Heathrow to Narita.*
- Two approaches
  - Fix the query and retrieve documents
  - Suggest corrected query option[s]

# Document correction

Especially needed for OCR'ed documents

- Correction algorithms are tuned for this: **rn/m**
- Can use domain-specific knowledge
  - E.g., OCR can confuse O and D more often ...
    - ... than it would confuse O and I (adjacent on the QWERTY keyboard, so more likely interchanged in typing).

But also: web pages and even printed material have typos

Goal: the dictionary contains fewer misspellings

# Isolated words correction

# Isolated word correction

Fundamental premise – **there is a lexicon** from which the correct spellings come

Two basic choices for this

A **standard** lexicon such as

- Webster’s English Dictionary
- An “industry-specific” lexicon – hand-maintained

The lexicon of the **indexed corpus**

- E.g., all words on the web
- All names, acronyms etc.
- (Including the mis-spellings)

# Isolated word correction

Given a lexicon and a character sequence  $Q$ , return the words in the lexicon closest to  $Q$

What's “closest”?

- Edit distance ([Levenshtein distance](#)) and LCS ([longest common subsequence](#))
- Weighted edit distance
- $n$ -gram overlap

# Edit distance

Given two strings  $S_1$  and  $S_2$ , the minimum number of operations to convert one to the other

Operations are typically character-level

- Insert, Delete, Replace, (Transposition)

E.g., the edit distance from ***dof*** to ***dog*** is 1

- From ***cat*** to ***act*** is 2 (Just 1 with transpose.)
- from ***cat*** to ***dog*** is 3.

Generally found by dynamic programming.

$$D(i, j) = \begin{cases} 0, & i = 0, j = 0 \\ i, & j = 0, i > 0 \\ j, & i = 0, j > 0 \\ \min\{ & D(i, j - 1) + 1, \\ & D(i - 1, j) + 1, \\ & D(i - 1, j - 1) + m(S_1[i], S_2[j]) \} \end{cases}$$

# Weighted edit distance

As above, but the weight of an operation depends on the character(s) involved

1. Meant to capture OCR or **keyboard errors**  
Example:  $m$  more likely to be mis-typed as  $n$  than as  $q$
2. Therefore, replacing  $m$  by  $n$  is a smaller edit distance than by  $q$
3. This may be formulated as a probability model

Requires weight matrix as input

# Edit distance to all dictionary terms?!

Given a (mis-spelled) query – do we compute its edit distance to every dictionary term?

- Expensive and slow
- Alternative?

How do we cut the set of candidate dictionary terms?

One possibility is to use  $n$ -gram overlap for this.

This can also be used by itself for spelling correction.

# *n*-gram overlap

Enumerate all the *n*-grams in the query string as well as in the lexicon

Use the *n*-gram index to retrieve all lexicon terms matching any of the query *n*-grams

**Threshold by number of matching *n*-grams**

- Variants – weight by keyboard layout, etc.

# Jaccard coefficient (IoU)

A commonly-used measure of overlap

Let  $X$  and  $Y$  be two sets; then IoU is  $|X \cap Y|/|X \cup Y|$

Equals 1 when  $X$  and  $Y$  have the same elements and zero when they are disjoint

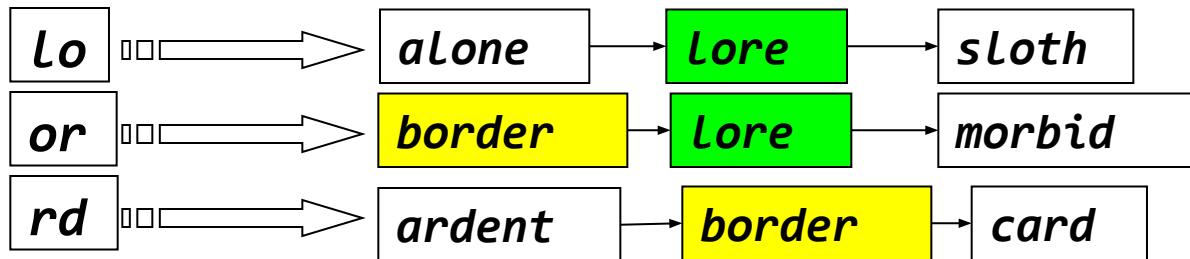
$X$  and  $Y$  don't have to be of the same size

Always assigns a number between 0 and 1

- Now threshold to decide if you have a match
- E.g., if  $\text{IoU} > 0.8$ , declare a match

# Matching trigrams

Consider the query ***lord*** – we wish to identify words matching 2 of its 3 bigrams (***lo, or, rd***)



Standard postings “merge” will enumerate ...

Adapt this to using Jaccard (or another) measure.

# Context sensitive correction

# Context-sensitive spell correction

Text: *I flew from Heathrow to Narita.*

Consider the phrase query “*flew form Heathrow*”

We'd like to respond:

Did you mean “*flew from Heathrow*”?

because no docs matched the query phrase.

# Context-sensitive correction

1. Retrieve **dictionary terms close** (in weighted edit distance) **to each query term**

Now try all possible resulting phrases with **one word “fixed” at a time**

- *flew from heathrow*
- *fled form heathrow*
- *flea form heathrow*

**Hit-based spelling correction:** Suggest the alternative that has lots of hits.

2. **Biword statistical approach**

Break phrase query into a **conjunction of biwords**.

Look for **biwords** that need **only one term corrected**.

**Enumerate** only phrases containing “common” biwords.

# General issues in spell correction

We enumerate multiple alternatives for “Did you mean?”

Need to figure out which to present to the user

- The alternative hitting most docs
- Query log analysis

More generally, rank alternatives probabilistically

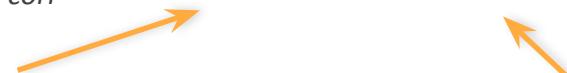
$$\operatorname{argmax}_{corr} P(corr \mid query)$$

- From Bayes rule, this is equivalent to

$$\operatorname{argmax}_{corr} P(query \mid corr) * P(corr)$$

Noisy channel

Language model



# Soundex: phonetic correction

# Soundex motivation

Class of heuristics to expand a query into **phonetic equivalents**

- Language specific – mainly for names
- E.g., *chebyshev* → *tchebycheff*

Invented for the U.S. census ... in 1918

# Soundex – typical algorithm

1. Turn every token to be indexed into a **4-character reduced form**
2. Do the same with **all query terms**
3. Build and **search** an **index** on the reduced forms (when the query calls for a soundex match)

# Soundex — part 1

1. Retain the first letter of the word.
2. Change all occurrences of the following letters to 0:  
    'A', 'E', 'I', 'O', 'U', 'H', 'W', 'Y'.
3. Change (similar) letters to digits as follows:
  - B, F, P, V → 1
  - C, G, J, K, Q, S, X, Z → 2
  - D, T → 3
  - L → 4
  - M, N → 5
  - R → 6

# Soundex — part 2

4. Remove all pairs of **consecutive digits**.
5. Remove all **zeros** from the resulting string.
6. Pad the resulting string with **trailing zeros** and return the first four positions, which will be of the form  
**<uppercase letter> <digit> <digit> <digit>**.

E.g., *Herman* becomes H655.

*H e r m a n n ?*

# Soundex and improvements

Even used by all databases, it is not very efficient in typo fixing. High recall, but very low precision.

Other phonometric algorithms exist:

Phonetic string matching (1996) =

= editorial distance + phoneme representation

Senc iu fo itenshn!

Vector model  
Distributive semantics  
Dimension reduction

Stanislav Protasov

# Agenda

- Vector interpretation of boolean query
- Distributive semantics
- Dimension reduction and LSA

Term document matrix

words\documents	Document1	document2	query term
cat	1	1	0
runs	1	1	0
behind	1	1	0
rat	1	0	1
dog	0	1	0

# Term-document matrix

TDM — describes the **frequency of terms** that occur in a collection of documents.

term-document matrix ... documents are the **columns** and terms are the **rows** (Wiki)



Term	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7
Information technology	0.77	0.55	0.45	0.13	0.14	0.15	0.15
Information systems	0	0	0.13	0.53	0.15	0.75	

In a *document-term matrix* ( $DTM = TDM^T$ ), rows correspond to documents in the **collection** and **columns correspond to terms**

- Column is a description (**vector**, BoW) of a document
- Row is a vector representation of a word
- **Sparse** for short texts



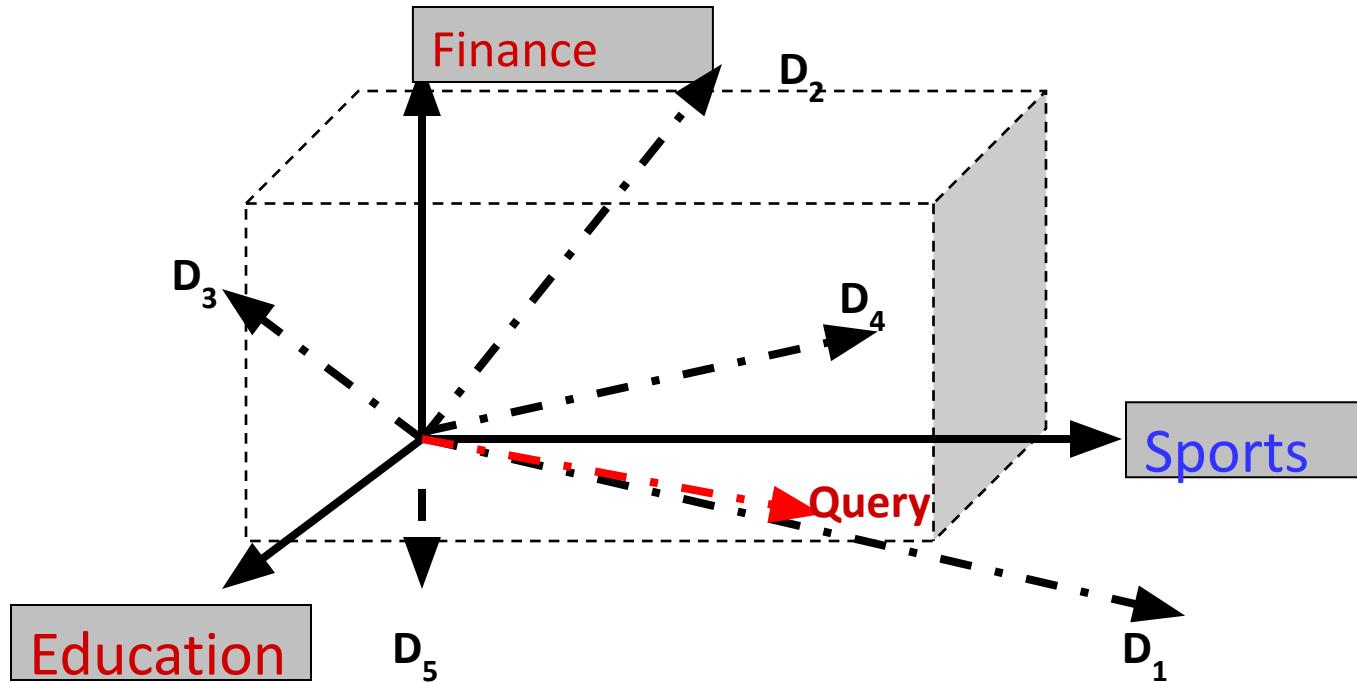
Venue	algorithm	cellular	game	hardwar	internet	mobil	network	search	secur	web
Keyword	2	8	0	24	0	5	0	2	1	1
algoritm	2	1	0	1	0	0	0	0	0	0
cellular	2	1	0	1	0	0	0	0	0	0
game	1	1	0	1	0	0	1	1	0	0
hardwar	1	0	1	4	0	18	0	0	1	0
internet	2	6	2	0	2	0	0	0	0	0
mobil	10	8	0	6	17	5	2	0	2	0
network	58	60	4	38	2	25	12	0	3	0
search	0	1	0	1	2	4	1	0	0	0
secur	4	4	29	5	1	12	3	0	4	0
web	0	2	0	3	3	1	13	0	2	0

# Vector space model

- Represents both *doc* and *query* by “concept vectors”
  - Each concept defines one **dimension**
  - $K$  concepts define a high-dimensional space
  - Element of vector corresponds to concept weight
    - E.g., for  $d=(x_1, \dots, x_k)^\top$ ,  $x_i$  is the “weight” of concept  $i$  (e.g. TF-IDF)
- Measures relevance approximation
  - **Distance** between the query vector and document vector in this concept space
  - relevance  $\approx$  similarity = 1 - distance
  - How can we define distance?

# VS Model: an illustration

- Which document is the closest to the query?

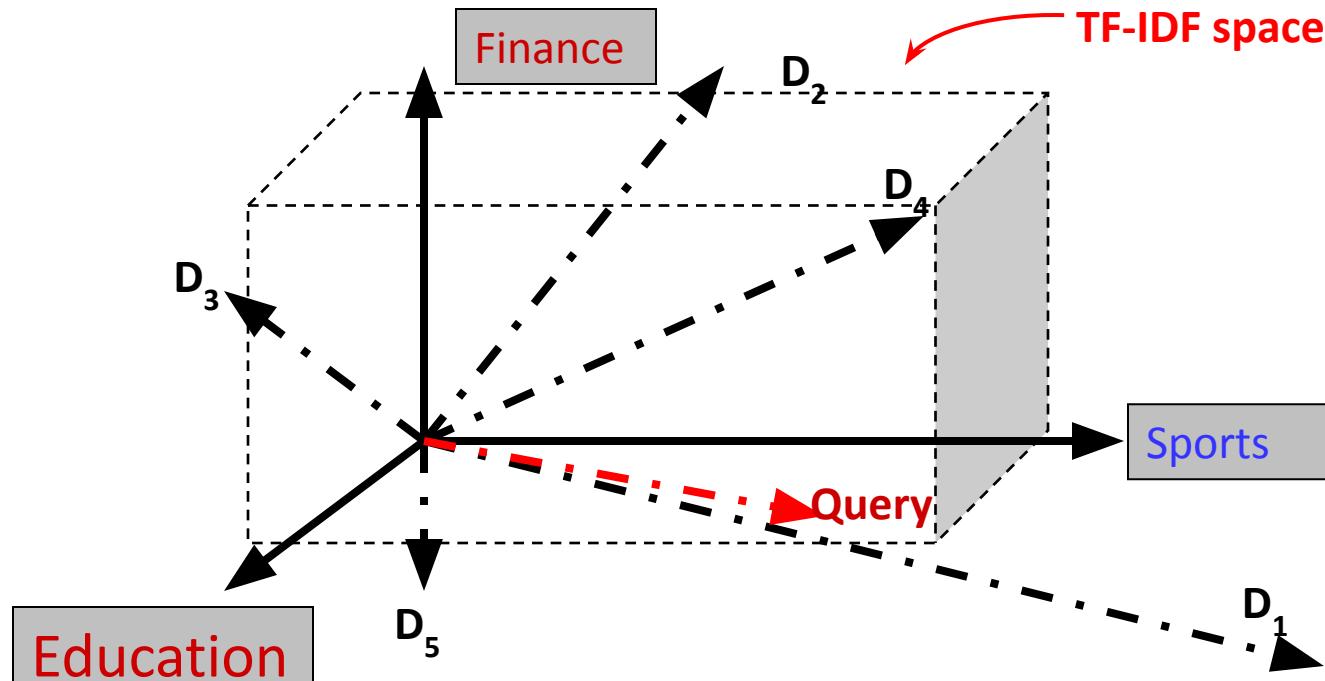


# What the VS model doesn't say

- How to **define**/select the “basic **concept**”
  - Concepts are assumed to be **orthogonal**
- How to assign **weights**
  - *Weight in a query* indicates importance of the concept for a query
  - *Weight in a doc* indicates how well the concept characterizes the doc
- How to define **distance** measure?

# How to define a good similarity measure?

- Euclidean distance?

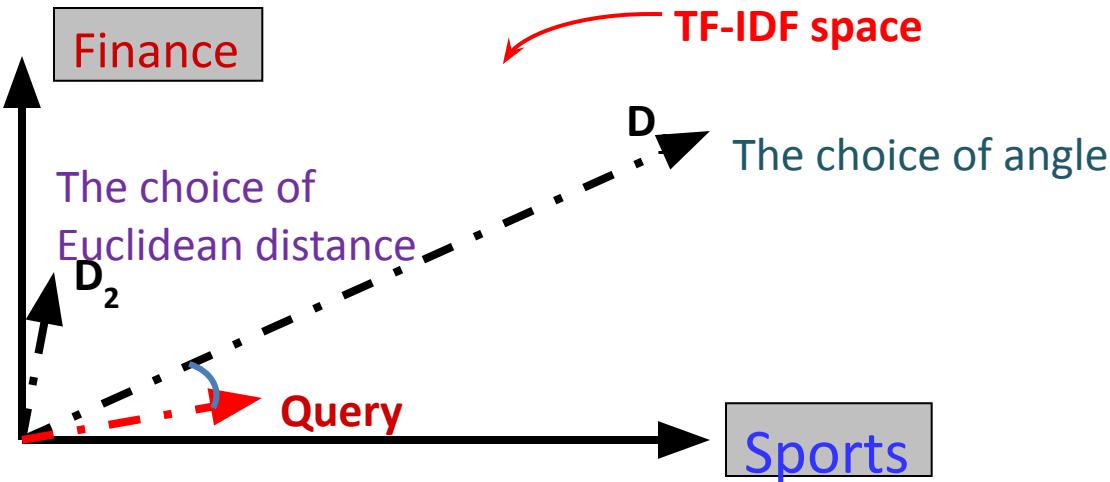


# From distance to angle

**Cosine similarity** – projection of one vector onto another

- $\pm 1$  if vectors are collinear
- 0 if vectors are orthogonal

$$\text{similarity} = \cos(\text{doc}, \text{query}) = \frac{\overrightarrow{\text{doc}} \cdot \overrightarrow{\text{query}}}{\|\overrightarrow{\text{doc}}\| * \|\overrightarrow{\text{query}}\|}$$



# Stop here!

1. We found a way, which allows to **represent any document** (even unseen) **as a vector**.
2. We introduced a **relevance metric** using a simple well-known mathematical concept - **cosine similarity**
3. Still **non-orthogonal** concepts
4. To measure similarity of 2 documents (or doc vs query) we need to do circa **100K arithmetic operations** with floating point numbers

# Reduce dimensions!

- Compression approach #1 — works great for **sparse databases**:

```
max_size = N  
doc_compressed[i % max_size] += doc[i] (or max, or =)
```

- Compression approach #2:
  - [Random projection](#)
  - Or even randomly remove some dimensions!
- Compression approach #3 — [latent semantic analysis](#):
  - [LDA](#), [PCA/SVD](#), GDA, ...
  - Embedding using encoder networks (BERT, doc2vec, DSSM, ...)

# Distributional semantics

Recall: **word** is just a **vector** in the vector space model

The **distributional hypothesis**:

- linguistic items with similar distributions have similar meanings
- words that are used and occur in the same contexts tend to purport similar meanings
- You shall know a word by the company it keeps (Firth, J. R. 1957)

Term document matrix			
words\documents	Document1	document2	query term
cat	1	1	0
runs	1	1	0
behind	1	1	0
rat	1	0	1
dog	0	1	0

# Notes on distributional semantics

“*Similar context*” and “*same distribution*” are not well-defined terms. It can go for bag of words model (TDM) or for near-context (as in word2vec).

Distributional hypothesis is a powerful model, which made to happen topic modelling, and all the ML-based NLP.

Consequence:

- If 2 **word**-corresponding **rows** from TDM **correlate**, we fail with orthogonality.
- But we can **infer orthogonal vector** from TDM!

**Hypothesis 2:** Maybe there is a **latent semantic space** of smaller dimension?

# Latent semantic analysis (patent)

Idea: search for low-rank approximation of TDM!

What does it mean? By now, query is  
(assume vectors are **normed**):

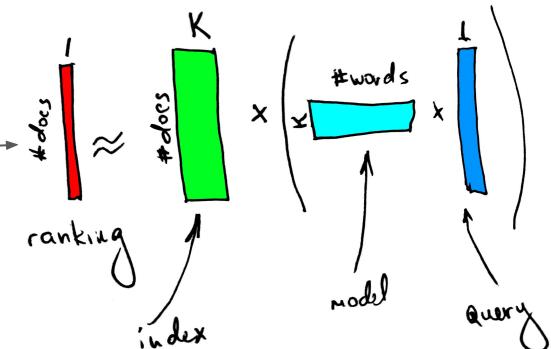
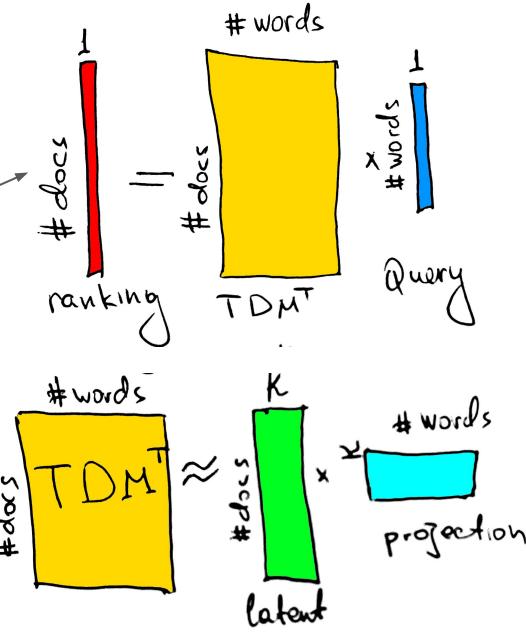
$$\text{Rankings} = \text{TDM}^T * Q_{\text{vector}}$$

What if

$$\text{TDM}^T = \text{LATENT\_MX} * \text{PROJECTION\_MX}$$

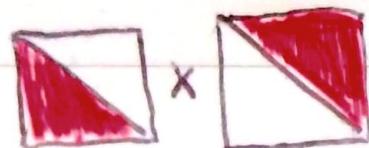
Then

$$\text{Rankings} = \text{LATENT\_MX} * [\text{PROJECTION\_MX} * Q_{\text{vector}}]$$

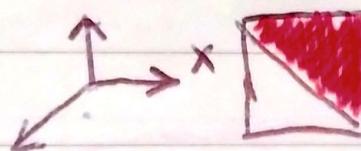


# Decompositions

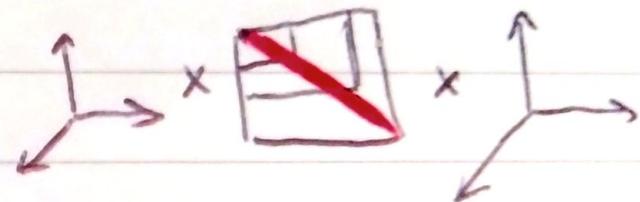
LU



QR



$$SVD = U\Sigma V^T$$

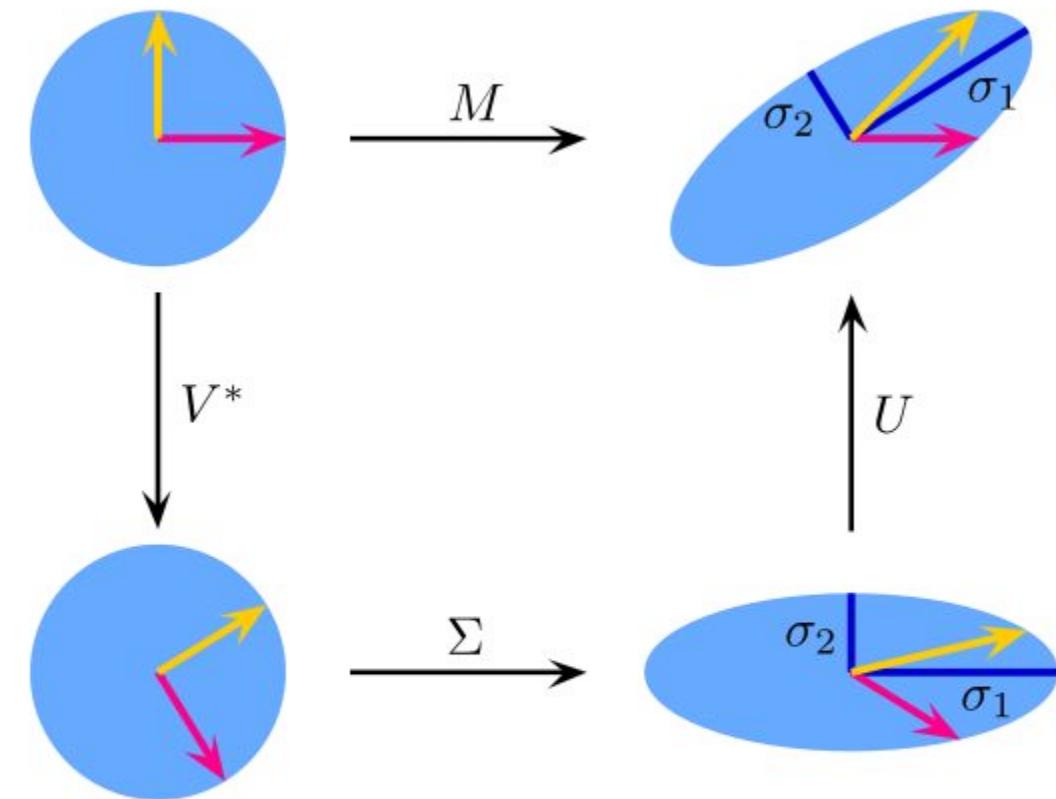


# SVD

$\mathbf{U}$  is eigenvectors for  $\mathbf{M}\mathbf{M}^T$

$\mathbf{V}^*$  is eigenvectors for  $\mathbf{M}^T\mathbf{M}$

$\Sigma$  is diagonal with square roots of non-negative eigenvalues of  $\mathbf{M}^T\mathbf{M}$



$$M = U \cdot \Sigma \cdot V^*$$

# Matrix approximation

$$M = U_R \Sigma_R V_R^T$$

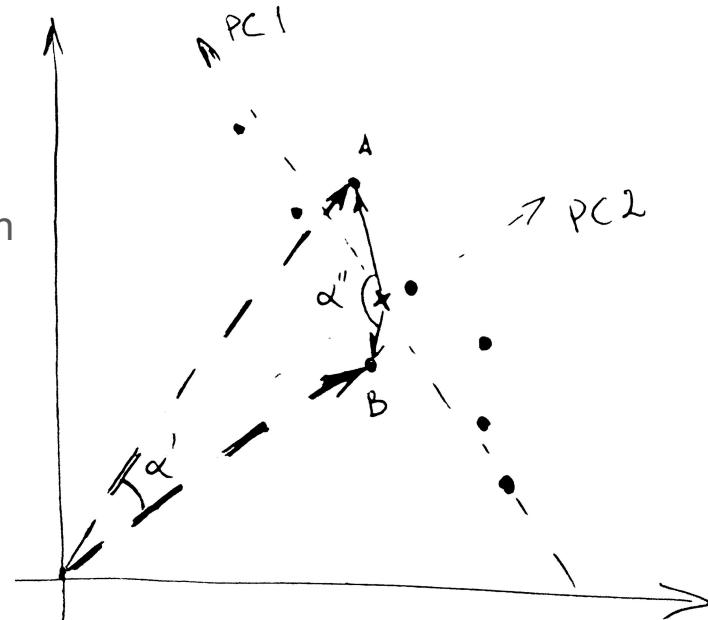
$$TDM^T = [U_R \Sigma_R]^* V_R^T$$

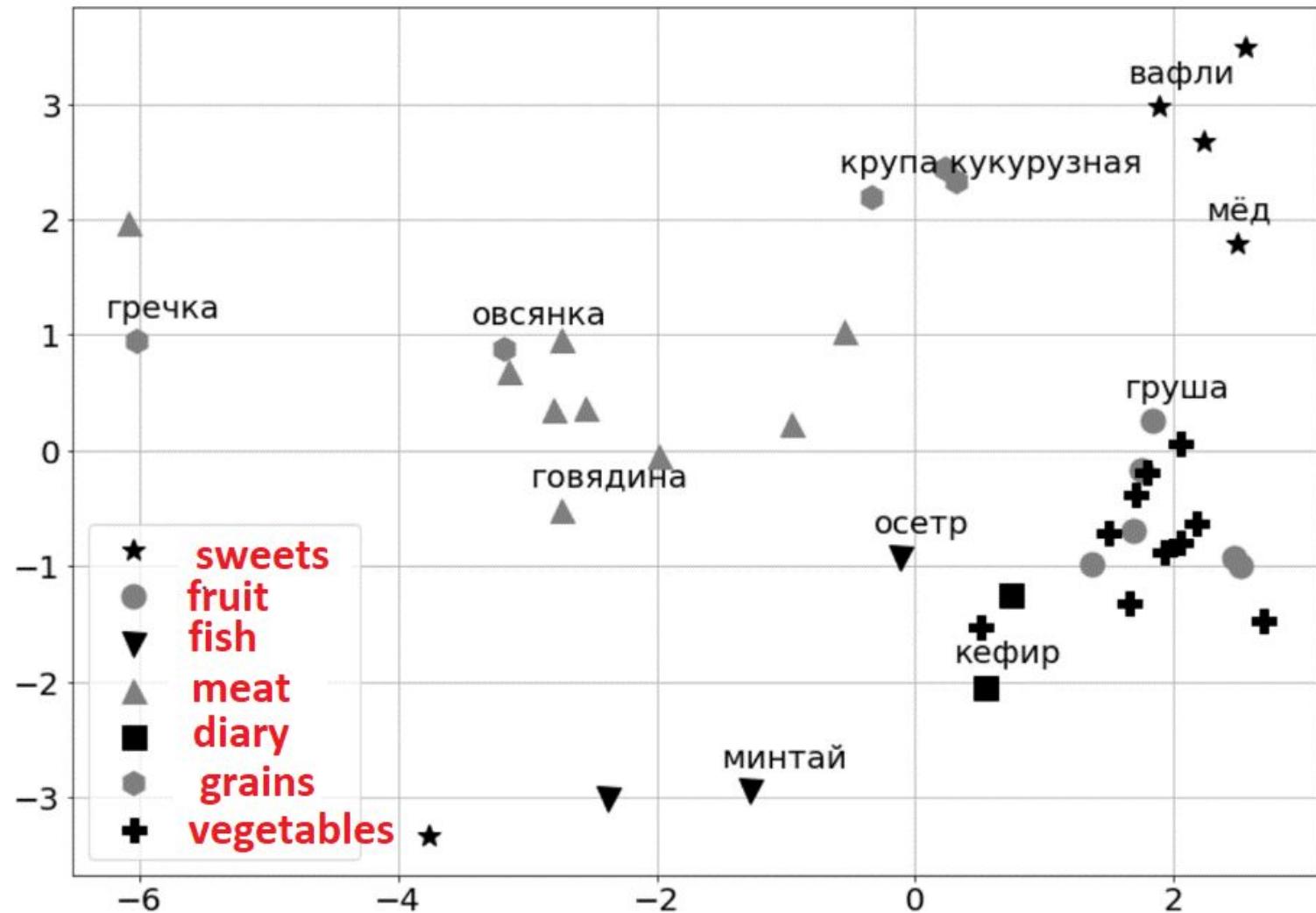
# Principal component analysis (similar idea)

... Convert a set of observations of **possibly correlated variables** (entities each of which takes on various numerical values) **into a set of values** of linearly uncorrelated variables ...  
(wiki)

... the **first principal component** has the **largest possible variance** ... and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components (wiki)

**NB:** Implemented with SVD, PCA requires **data centering** first, which affects the **cosine metric**.





# Stop Here!

1. **Vector space** model is cool, but (1) TDM is sparse (2) concepts are not orthogonal
2. **Distributional hypothesis** gives an insight: words correlate and their distribution defines semantics
3. **Latent semantic analysis** says: yes, and we know that there is a small-dimensional latent space for semantics. TDM is just a **linear projection**
4. **SVD** says: mmmm... We know how this latent space should look like!  
Orthogonal features + decreasing variance

# Reading

- The Book — chapter 6.2-6.5
- All links in this presentation

# Vector space modelling with ML

Stanislav Protasov

# Agenda

- LSA — what important is missing?
- ANNs solving embedding task
  - word2vec, doc2vec
  - DSSM
  - Transformers

# LSA critics

**Speed issue.** Even optimized SVD is slow and requires memory and CPU time

- Fast Randomized SVD (facebook)
- Alternating Least Squares (ALS). Distributed and streaming versions

**Model issue.** PCA assumes **normal** data distribution, but life is complicated; SVD preserves **angles**, but angle != semantic similarity. Both dimension reduction methods are **global**.

- **pLSA.** Statistical independence (**any distribution**) vs linear orthogonality.

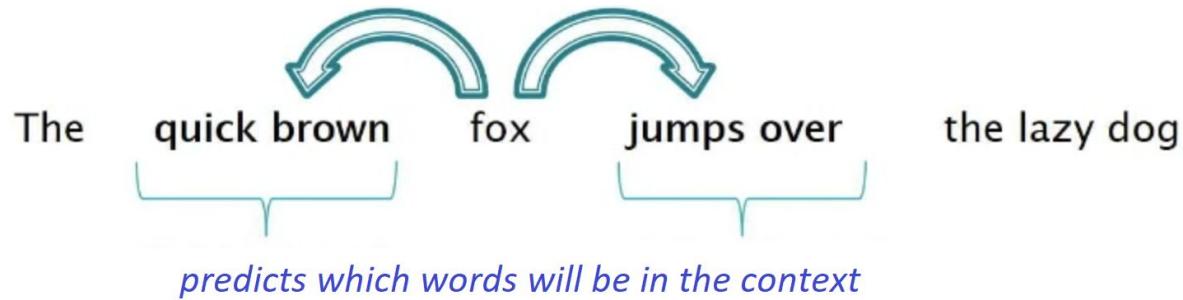
What about **adding new** words/texts?

Can we take some model and don't care about distributions, statistics, memory and so on?

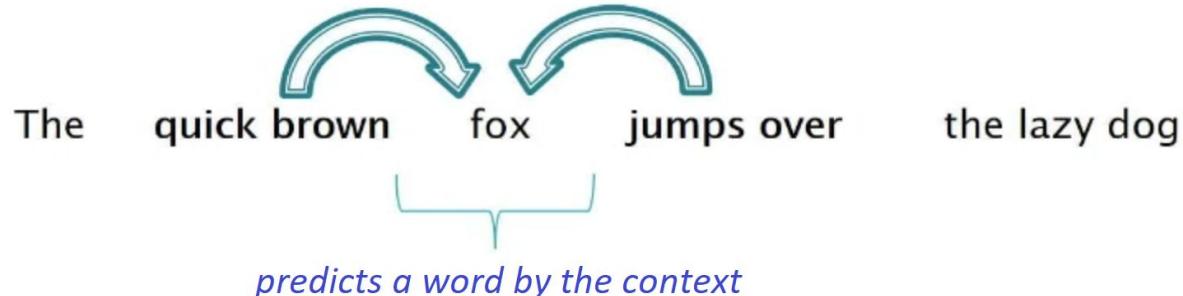
# word2vec (2013) - group of methods

Do not compute, predict!

Skip-grams:



CBoW:



# CBoW - Continuous bag of words

BoW - multiset of objects disregarding order (works for images, texts, ...)

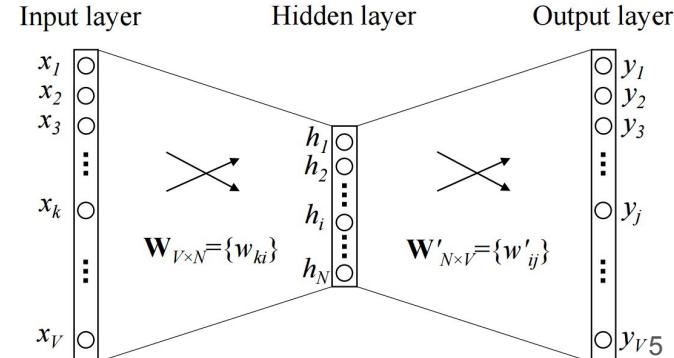
CBoW - continuous sample of text (window)

Input - one-hot **context** encoding (dict-size vector)

Output - overall **collection distribution** (dict size vector)

Activation (softmax) models a **likelihood**  $p(w|c, \theta)$

**Where is embedding?**

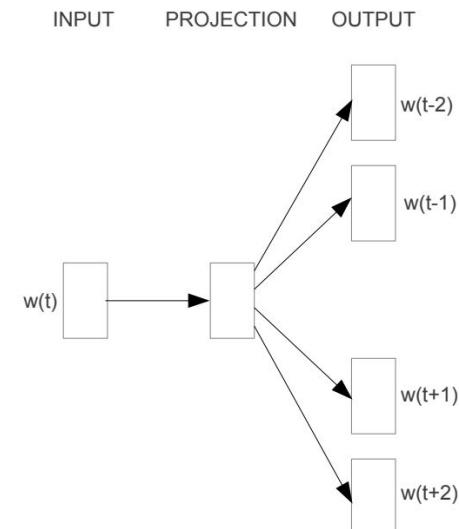


# Continuous skip-gram model

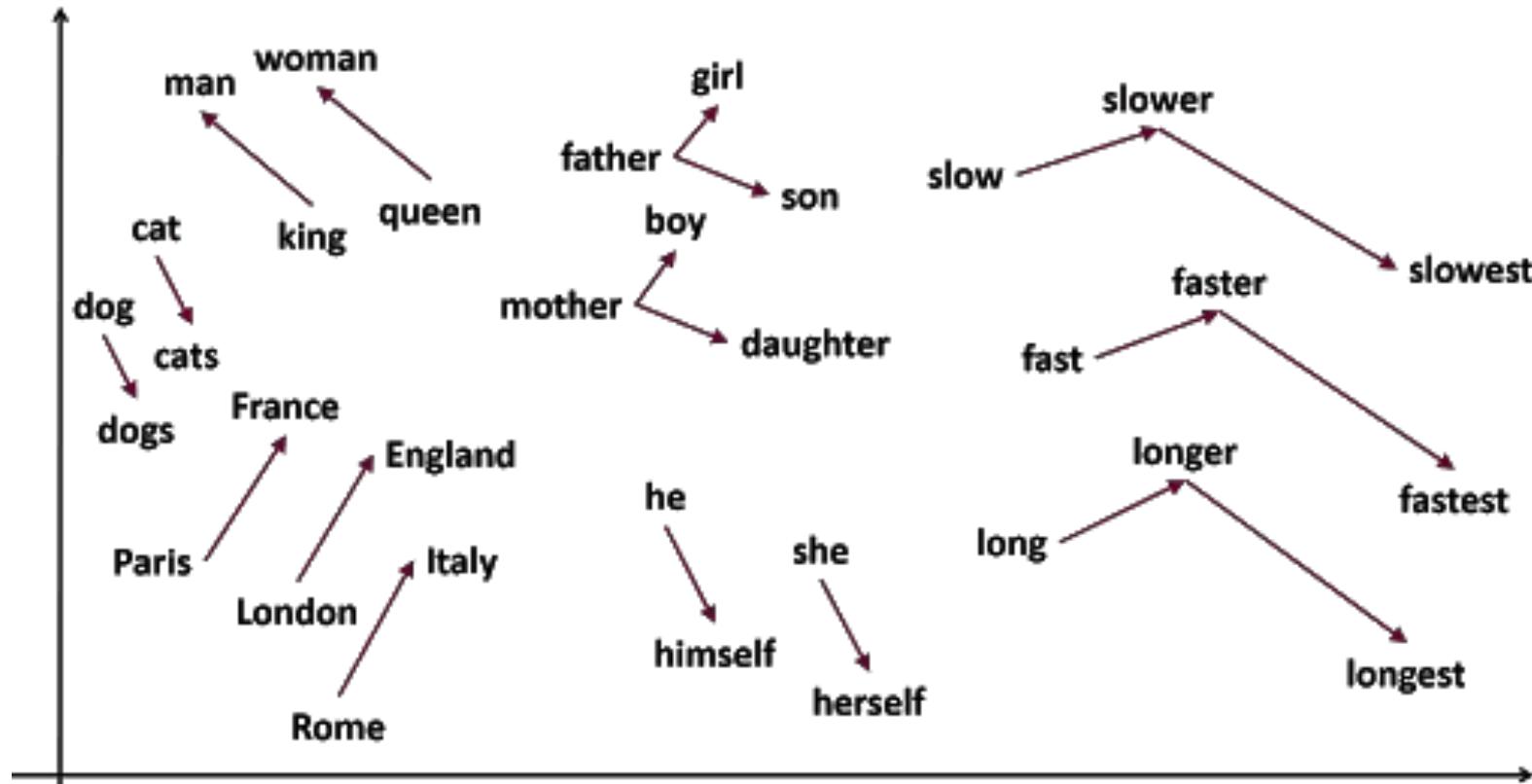
“... instead of predicting the current word based on the context, it tries to **maximize classification** of a word based on **another word in the same sentence**” (paper)

**increasing the range improves quality** of the resulting word vectors, but it also increases the computational complexity

...we give less weight to the **distant words** by **sampling less** from those words in our training examples.



# Bonus: vector space arithmetics



# Training and quality

... we used three training epochs with **stochastic gradient descent** and **backpropagation**. We chose starting learning rate 0.025 and decreased it linearly, so that it approaches zero at the end of the last training epoch.

... there are 8869 **semantic** and 10675 **syntactic** questions.

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

# word2vec (and almost everyone's) problems

OOV - **out of vocabulary** or even underrepresented words.

- *today is approached with subword tokenization*

*Grammar, abbreviations, forms* and *homographs* — out of scope. (Paper don't discuss lexers or stemmers)

Training depends on N (context size)  $\times$  D (dictionary).

Still works with **words, not paragraphs** or sentences.

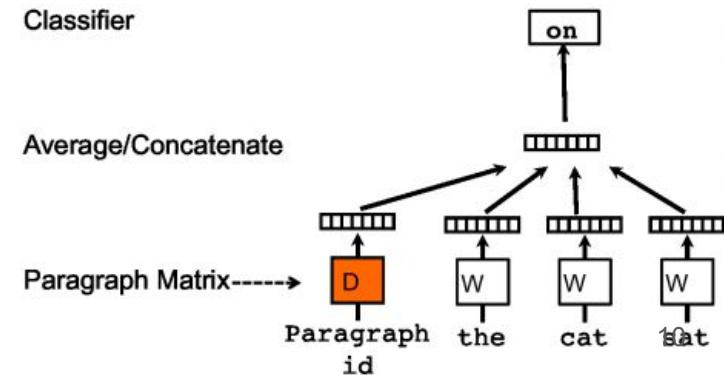
# doc2vec (Paragraph Vectors)

... we concatenate the paragraph vector with several word vectors from a paragraph and predict the following word in the given context. ... paragraph vectors are unique among paragraphs, the word vectors are shared.

**No syntax:** using a parse tree to combine word vectors, has been shown to work for only sentences because it relies on parsing.

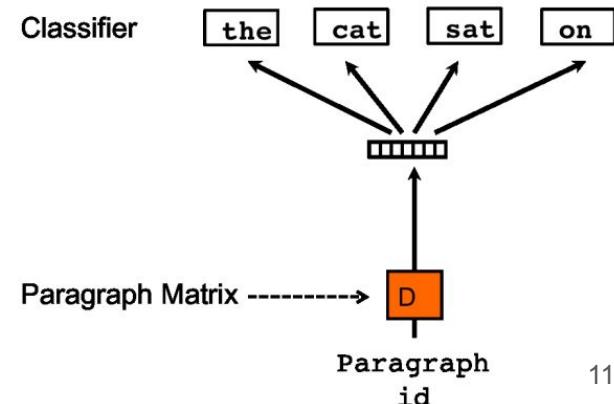
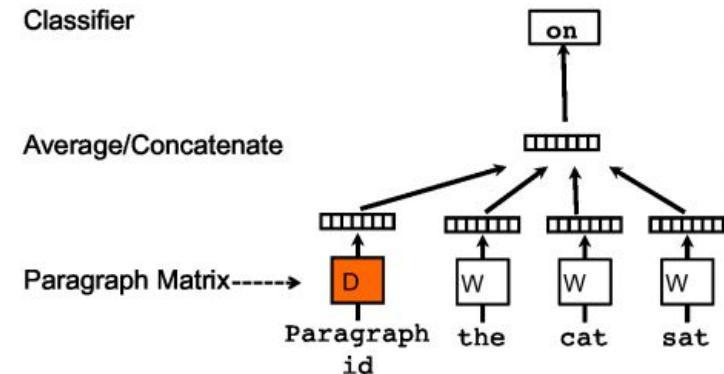
The **paragraph token** can be thought of as **another word**. Paragraph vector length can be different.

“the inference stage” to get paragraph vectors D for new paragraphs (never seen before) by **adding more columns in D and gradient descending on D** while holding W, U, b fixed



# Details

1. Paragraph Vector - **Distributed Memory**  
(PV-DM) - concatenate paragraph and word vectors
2. Paragraph Vector - **Distributed Bag of Words**  
(PV-DBOW) - ignore the context words in the input, but force the model to predict words randomly sampled from the paragraph in the output. Sample a text window, then sample a random word from the text window and form a classification task given the Paragraph Vector



# Training and quality

- SGD
- NB Paragraph vector inference requires running GD!
- Sentiment analysis (5-class, 2-class) classification
  - Special characters such as ,.!? are treated as a normal word

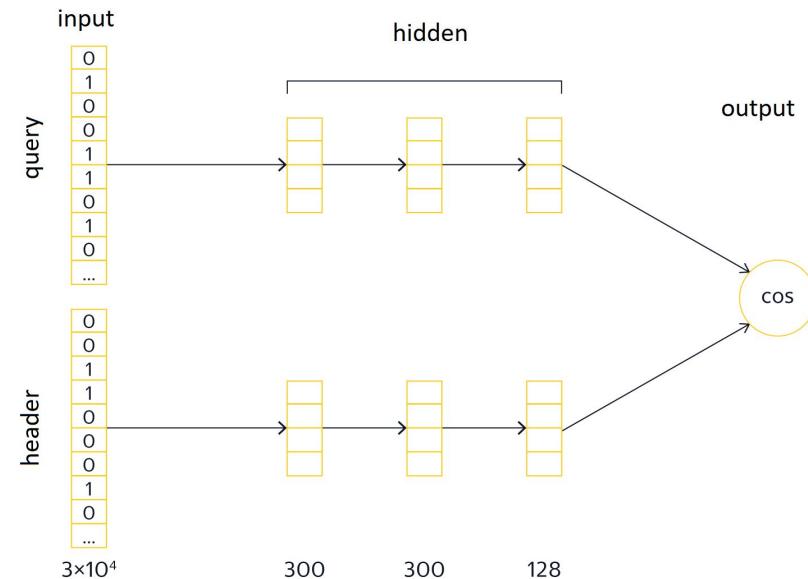
Model	Error rate
BoW (bnc) (Maas et al., 2011)	12.20 %
BoW ( $b\Delta t'c$ ) (Maas et al., 2011)	11.77%
LDA (Maas et al., 2011)	32.58%
Full+BoW (Maas et al., 2011)	11.67%
Full+Unlabeled+BoW (Maas et al., 2011)	11.11%
WRRBM (Dahl et al., 2012)	12.58%
WRRBM + BoW (bnc) (Dahl et al., 2012)	10.77%
MNB-uni (Wang & Manning, 2012)	16.45%
MNB-bi (Wang & Manning, 2012)	13.41%
SVM-uni (Wang & Manning, 2012)	13.05%
SVM-bi (Wang & Manning, 2012)	10.84%
NBSVM-uni (Wang & Manning, 2012)	11.71%
NBSVM-bi (Wang & Manning, 2012)	8.78%
Paragraph Vector	7.42%

Go deeper?



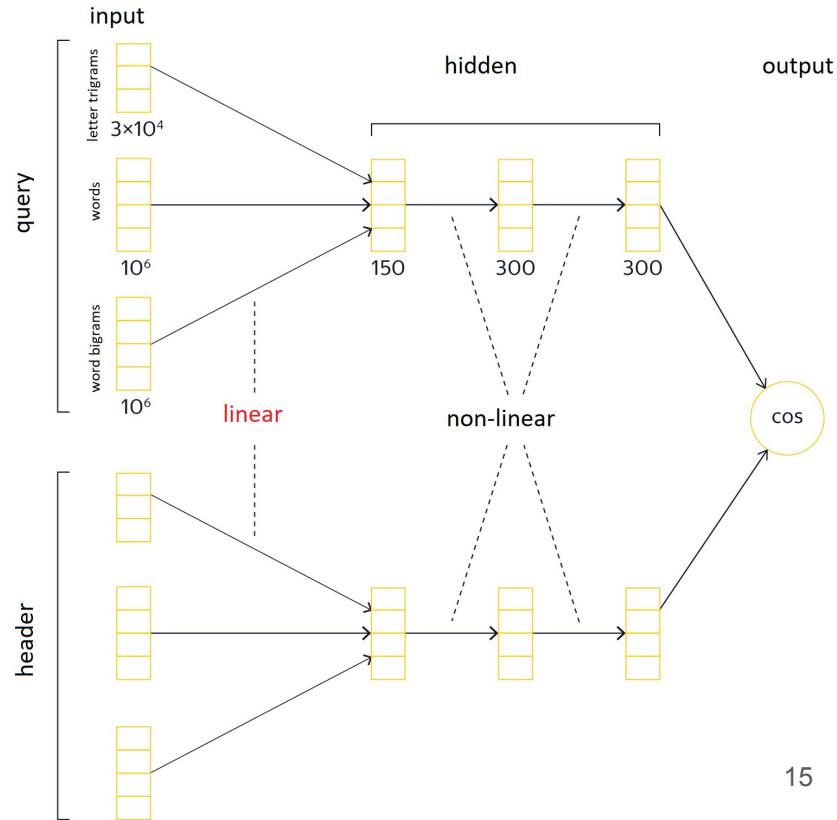
# DSSM - Deep Structured Semantic Model (by MS)

- Original architecture:
  - Trained to predict cosine similarity
  - Uses bag of **letter trigrams**
- Important:
  - Initially created for **search** (uses specific metric)
- Problem:
  - Relatively **small input size** ( $33^3/26^3$  of trigrams) for deep network
- Training:
  - Positive - clicked headers
  - Negative - shown but not clicked
  - **Not necessary relevance!**



# DSSM update by Yandex

- Input layer:
  - Trigrams
  - **+1M of words**
  - **+1M of word bigrams**
- Training:
  - Failed on random negatives
  - Failed on fake negatives
  - hard negative mining: similar to GANs, but simpler, as network fights itself
  - Another target: **dwelltime**



# BERT (Bidirectional Encoder Representations from **Transformers**), YATI

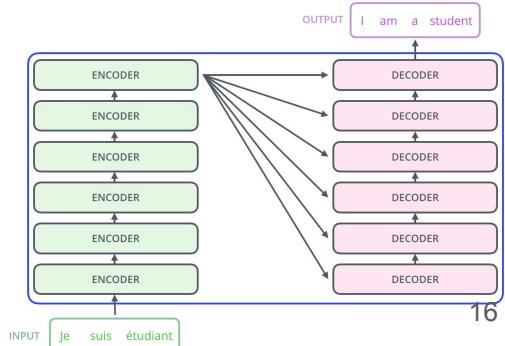
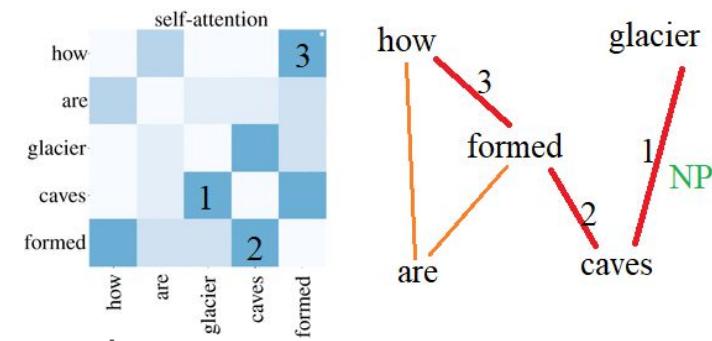
Created to learn **language model** — and to solve general language tasks

Attention and self-attention (~syntactic tree)

Modes:

- Trained to predict 15% of masked words
- Also trained to predict logical connections between phrases

*“The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training”*



# Reading

Papers and articles (links in presentation) on this topic

# Approximate nearest neighbours search

Stanislav Protasov

# Agenda

- ANNS (not ANNs)
  - Clustering and IVF
  - Proximity graphs (NSW, HNSW)

# Before we start...

What's wrong with inverted index in terms of data structure?

Do you know the difference:

- $O(f(N))$ ,
- $O_A(f(N))$ ,
- $E(f(N))$ ?

# Approximate Nearest Neighbours Search

# Approximation for k-NN search

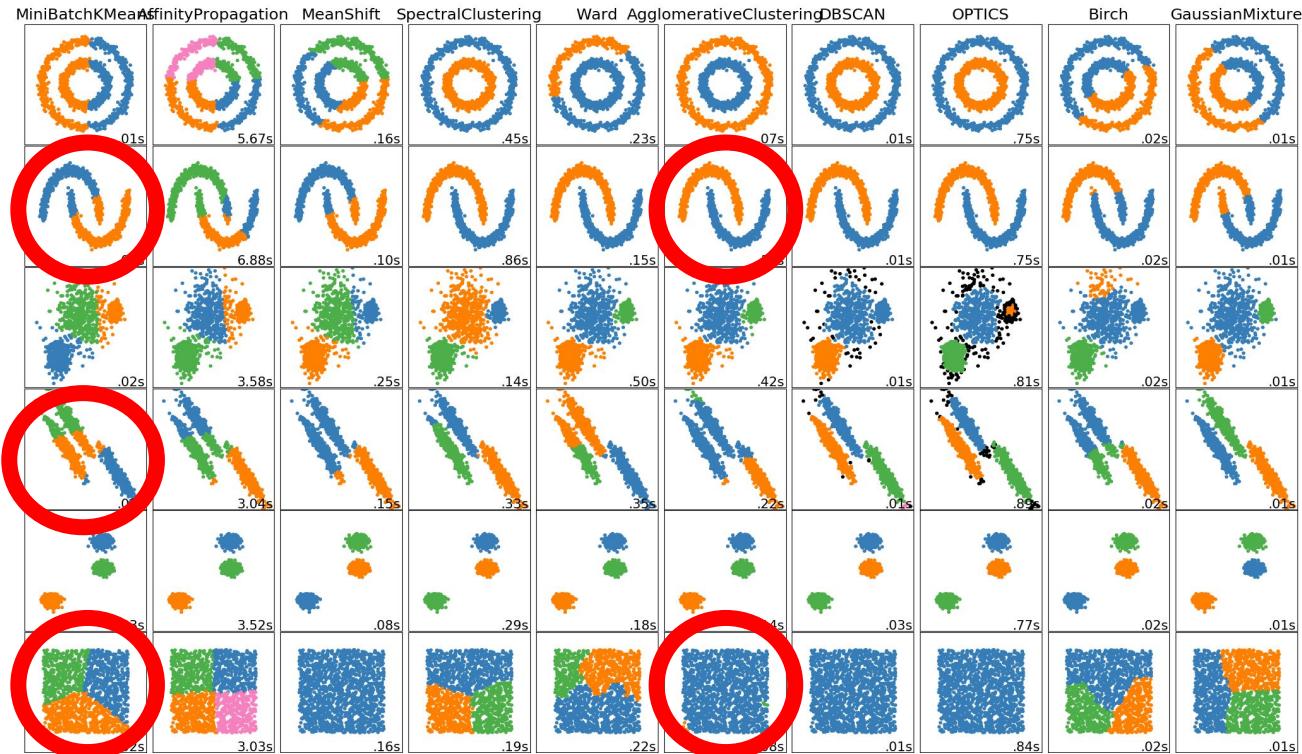
1. Pre-select  $k*c$  elements from approximate neighbourhood (~pre-ranking set). \* Practical example: **Recall@1000  $\geq 0.875$**
2. Then select and re-rank relevant ones.

How to:

- Locality sensitive hashing
- **Search trees and supporting data structures**
- **Vector compression, clustering, inverted indexing**
- **Proximity graphs**

# Hierarchical clustering and Inverted index revised

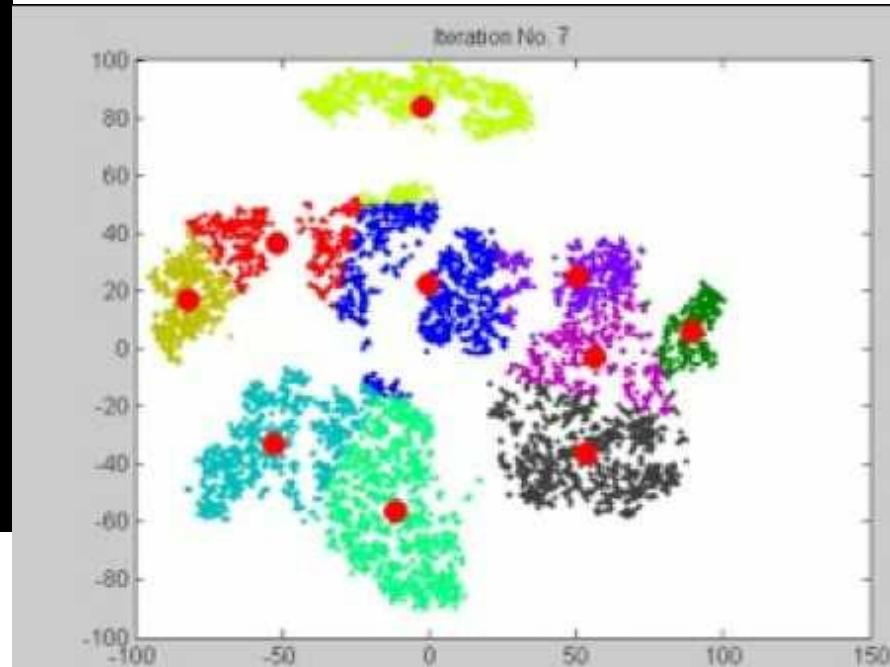
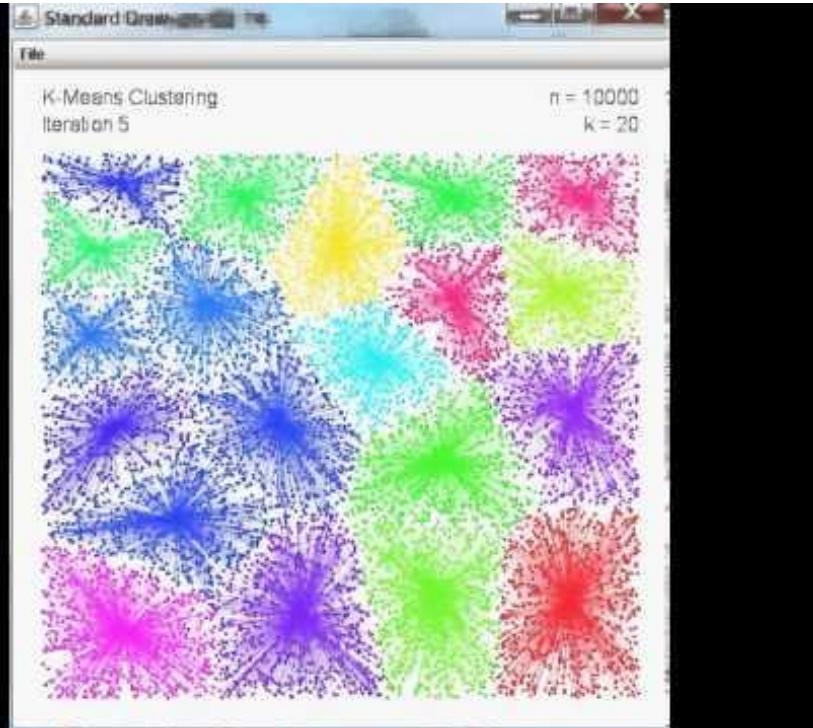
# How clustering approaches differ?



# Linkage criteria

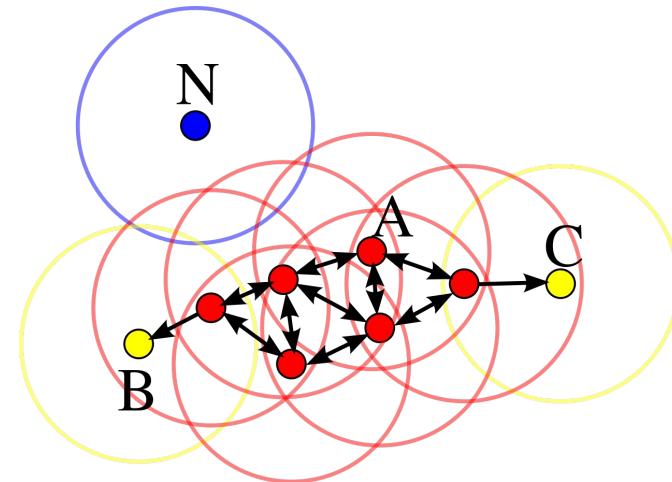
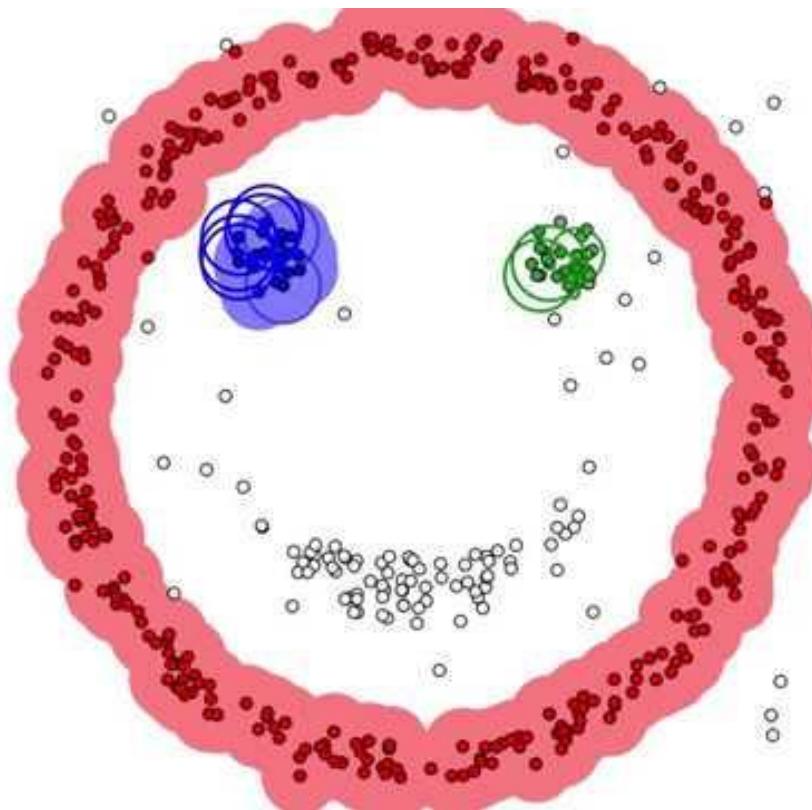
- **Single linkage** (smallest distance) ~ DBSCAN
- Complete linkage (maximum distance)
- Minimum energy (variance grows slowly in we merge)
- Average distance and **centroid-based approaches** — kMeans

# K-Means



# DBScan

Density-based spatial clustering of applications with noise.



# Why do we cluster?

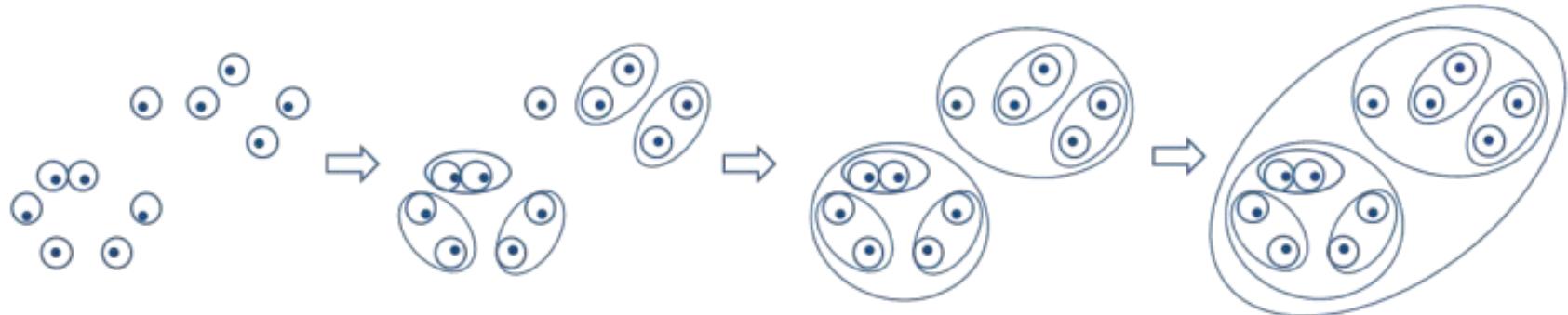
For a flat list we run  $O(N)$  comparisons to find kNN

For  $\sqrt{N}$  similar\* clusters we can pick one closest\*\*  
for  $O(\sqrt{N})$  and find  $k$  NNs\*\*\* in  $O(\sqrt{N})$ .

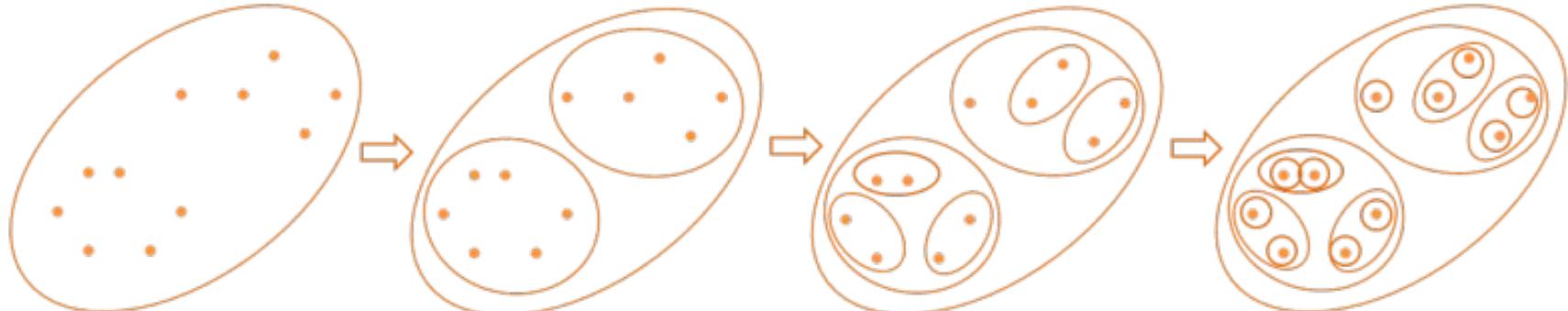
For two layers of  $^3\sqrt{N}$  ...

# How do we cluster?

Agglomerative Hierarchical Clustering

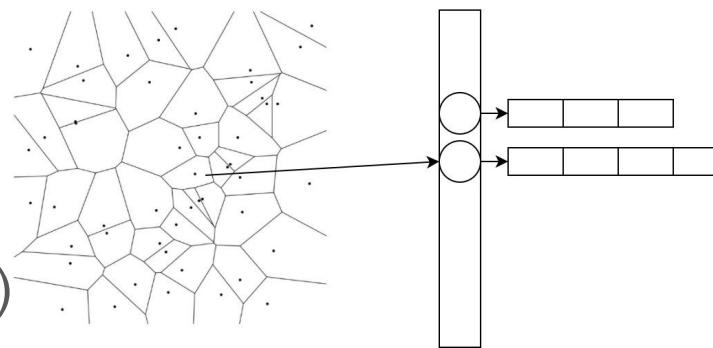
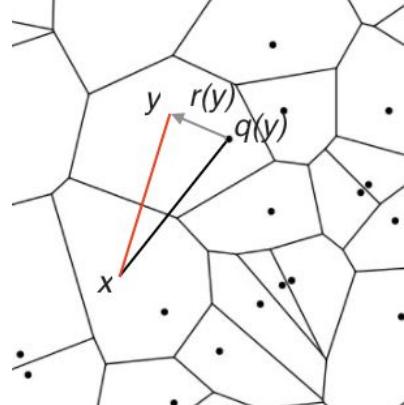


Divisive Hierarchical Clustering

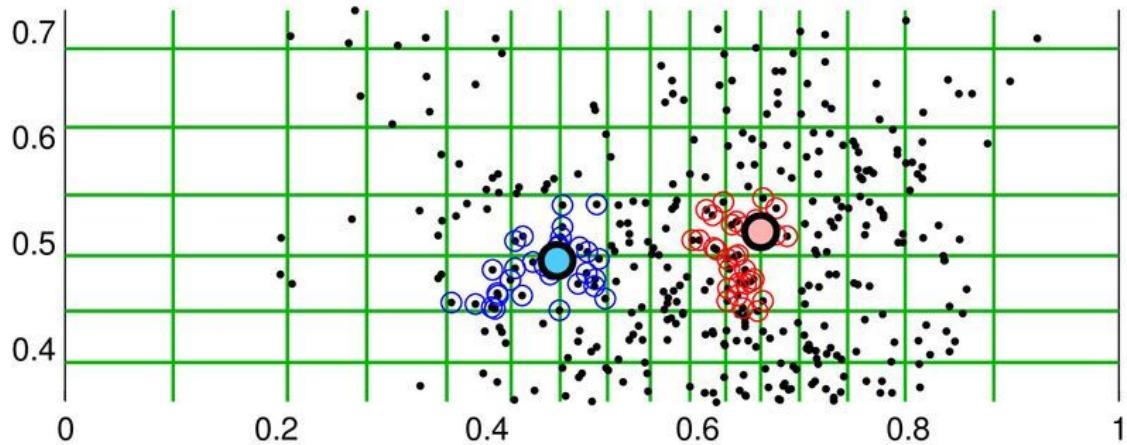
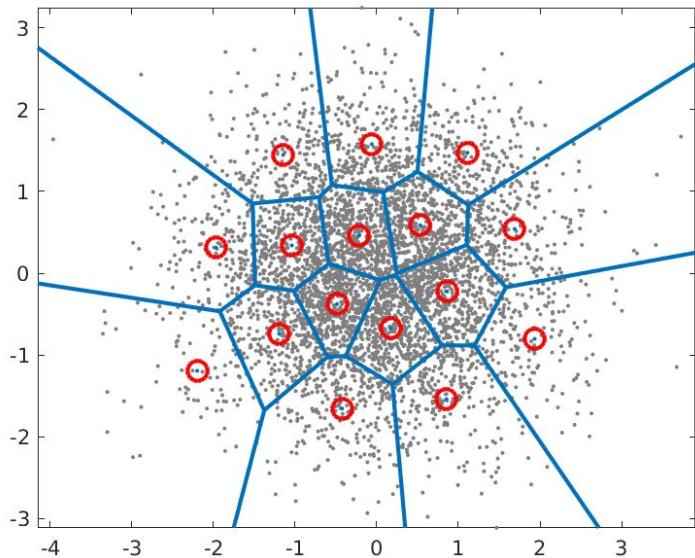


## Revised IVF. [FAISS](#) (Facebook AI similarity search)

- Uses [Voronoi diagram](#) clusters (kMeans).  
Vectors are approximated with **centroids (VQ)**  
And compressed with  
**scalar quantization (SQ)**
- Build **inverted index** for points in clusters
- Vector compression: product quantizer (**PQ**)
  - Split  $\mathbf{R}^{128}$  into 8 groups of 16 floats
  - Perform 256-means clustering of these “sub-vectors” and encode with 1 byte each



# Vector Quantization and Product Quantization

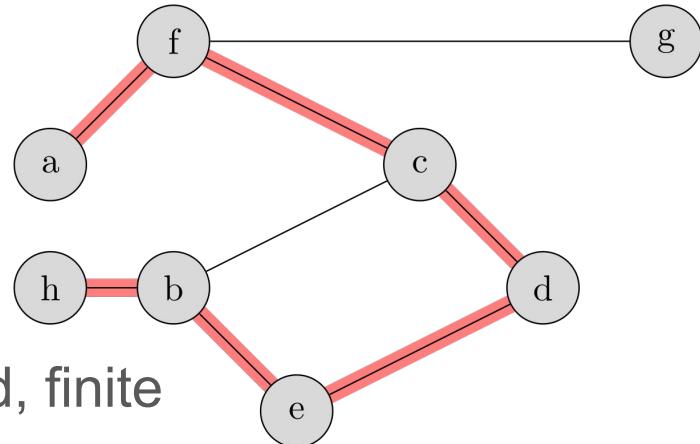


<https://wiki.aalto.fi/pages/viewpage.action?pageId=149883153>

<https://arbabenko.github.io/MultiIndex/>

# Approach #2. Proximity graphs

# Graphs cheat sheet



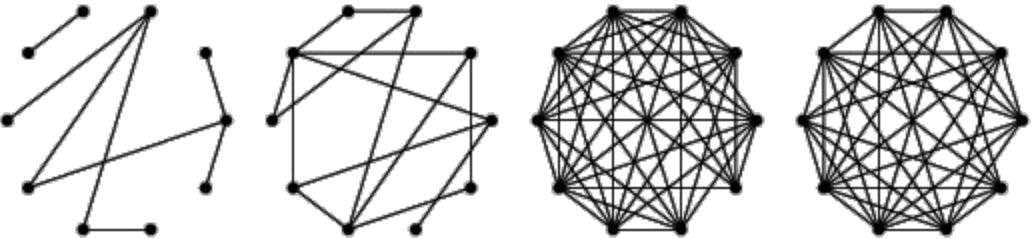
**Graph -  $G = (V, E)$** , can be weighted, directed, finite

**[Simple] path** - sequence of vertices and edges

**Degree of vertex** - number of incident edges

**Graph diameter** - longest shortest path between a pair of vertices

## Random graph



Some random process (uniform, Gaussian, ...) generates edges.

Almost every graph in the world. *Previously* considered as a model for social networks.

Small average shortest path - which is **good** for search.

Small clustering coefficient (defines how close are neighborhoods to cliques) - which is **bad** for **NN search**.

$$C(v) = \frac{e(v)}{\deg(v)(\deg(v)-1)/2}$$

$$\tilde{C} = \frac{1}{N} \sum_{i=1}^N C(i)$$

# Regular graphs

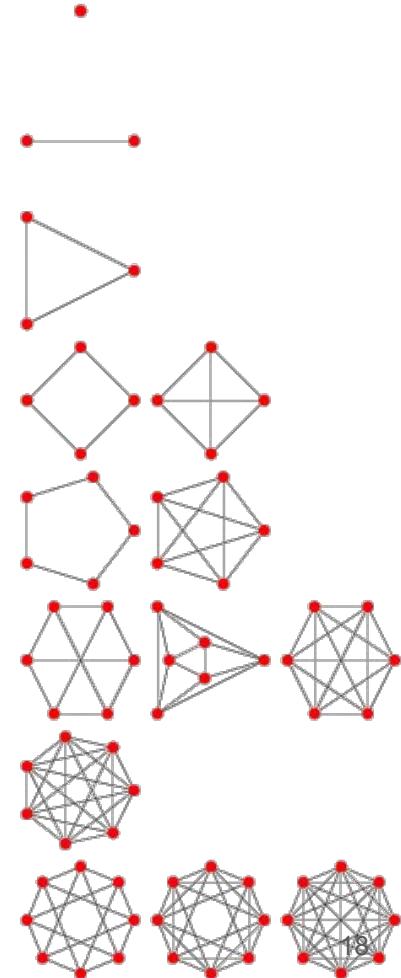
$K$ -regular graph is a graph with  $\deg(v) = K$  for any  $v$ .

Used to model big homogeneous networks.

Can also be random (as there are multiple  $K$ -regular graphs on the same size)

Big diameter - which is **bad** for **search**

Big clustering coefficient - which is **good** for **NN search**



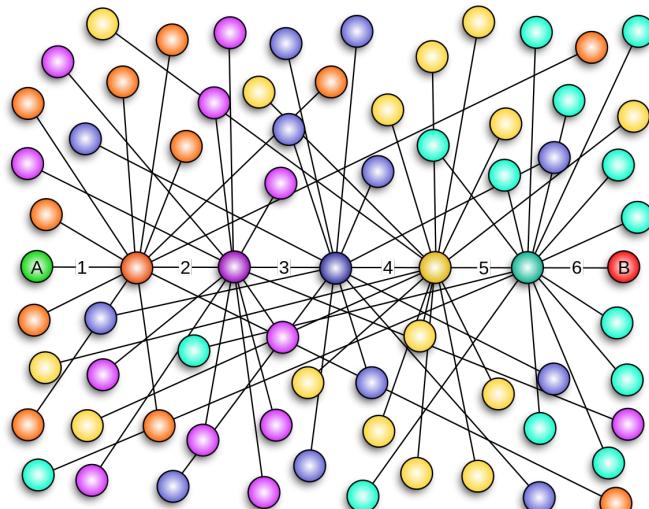
# Small World experiment by Stanley Milgram, 1967

Initially it was considered, that social graph is kind of regular.

Experiment discovered (even with some questions to method) that even graph is **highly clustered, average path length is small.**

Was a basis for 6 handshakes rule.

New type of graphs was suggests:  
small world networks.



## Small world network

Most vertices are not neighbours (small degree means *sparse graph*).

Nevertheless, small number of hops needed to reach any other node.

Typical path length  $L$  between 2 random nodes (of  $N$ ):  $L \propto \log N$

Many real world networks are like this: internet, wiki, social graphs, power grids, brain cells. Although not all real networks like SW: many-generation networks, classmate graphs.

[Watts–Strogatz model](#) and [Kleinberg model](#) are how we describe and build SW networks

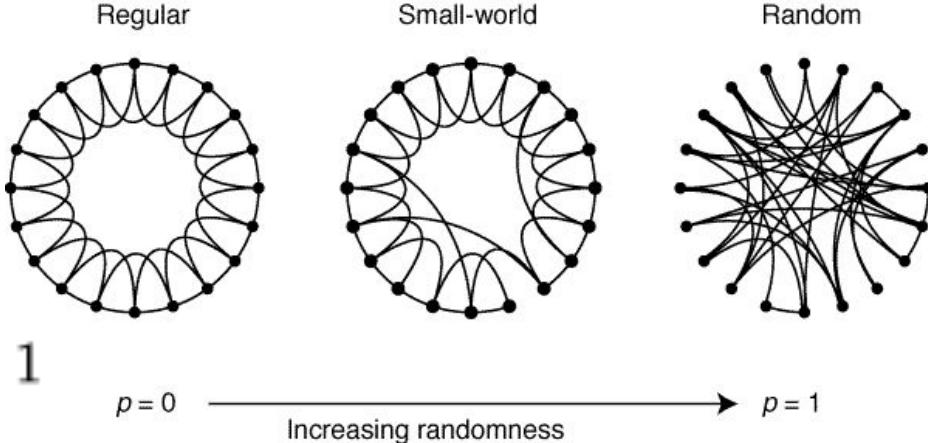
## Watts–Strogatz model

Given  $N$  nodes and  $K$ -”regularity”

(average degree  $K$ )  $N \gg K \gg \ln N \gg 1$

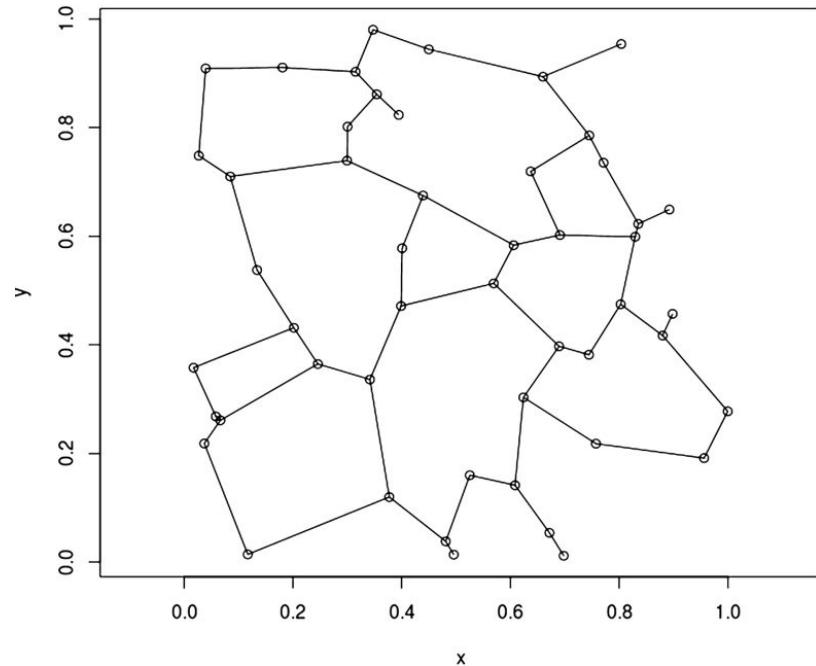
Given parameter  $p$  from  $[0, 1]$ .

- 1) Construct a regular ring lattice.
- 2) take every edge connecting **vertex** to its  **$K/2$  rightmost neighbors**, and rewire it with probability  $p$ . Rewiring is done by **replacing destination** with vertex  $k$  (chosen **uniformly** at random from all possible nodes while avoiding self-loops and duplication).



# Proximity graphs

A proximity graph is simply a graph in which two vertices are connected by an edge if and only if the vertices satisfy particular geometric requirements.



## Navigable small world networks

Idea is similar to [skip-lists](#).

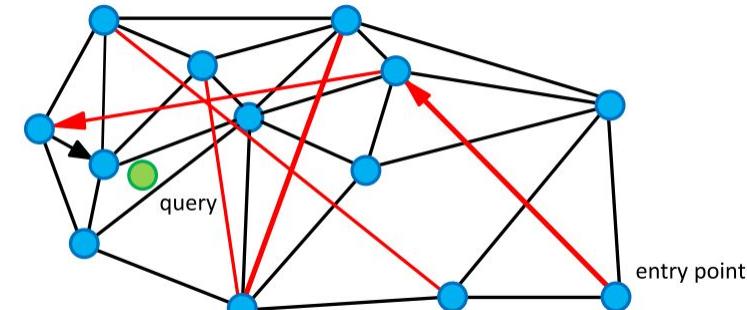
We can also measure **distance** (e.g. dot product, Euclidian,  $L_k$ -norm, Hamming, Levenshtein, ...) between *query* and *current vertex*. Originally *Delaunay graph* needed to converge for exact search, but ANNS allows other small-world graphs.

Building:

1. One-by-one insertion via kNN search. Distant edges are created in the beginning.

Search:

1. Perform greedy search. Move to the neighbour vertex **closest to query**
2. Update NN set on each step until it converges



# Hierarchical navigable small world (github)

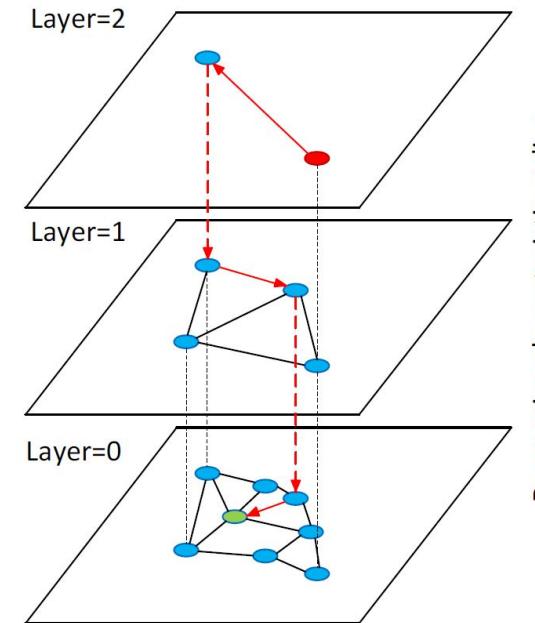
Layer 0 holds complete NSW network

Ideas:

- Better **start search** from a node with **high degree**
- Higher layer has longer links (skip-list!)
- Decrease layer size exponentially

**Highlight:**

- 1) **search procedure requires only  $\text{dist}(u, v)$  function**
- 2) **No embedding, hyperplanes, centroids of whatever needed**



To read

An Introduction to Proximity Graphs

Efficient and robust approximate nearest neighbor search  
using Hierarchical Navigable Small World graphs

How can proximity graphs benefit in classification and hybrid  
memory

# Forests of search trees

Stanislav Protasov



# Agenda

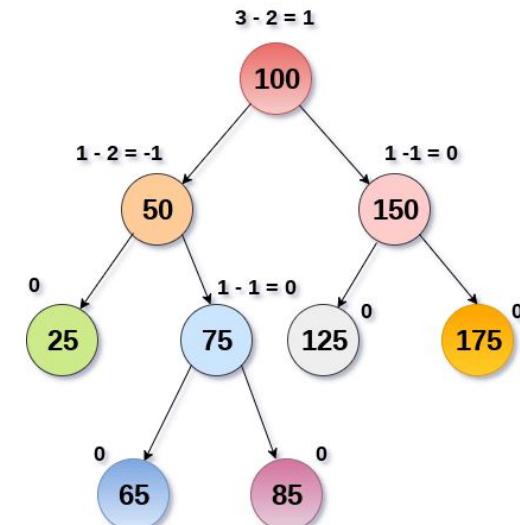
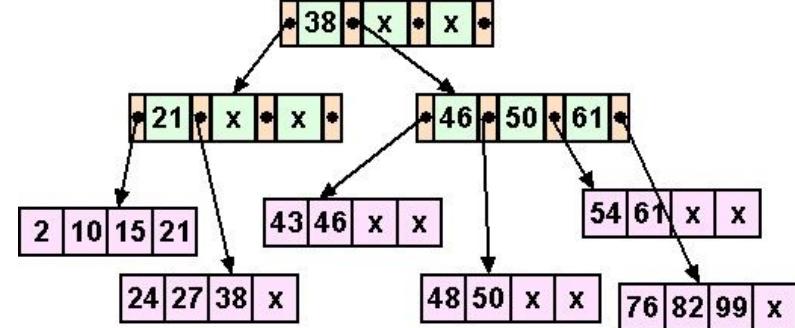
ANNS with trees:

- Search trees
- Quad trees
- KD-trees and Ball Trees
- Annoy
- And some others

# Search Trees

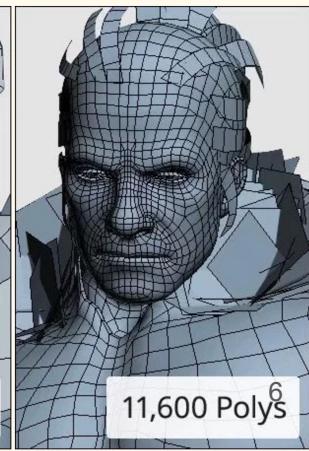
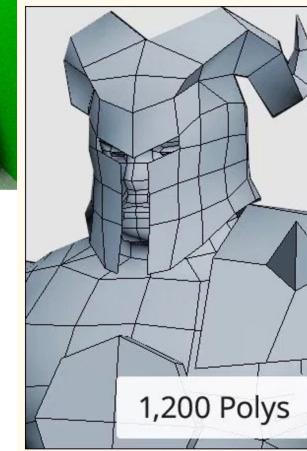
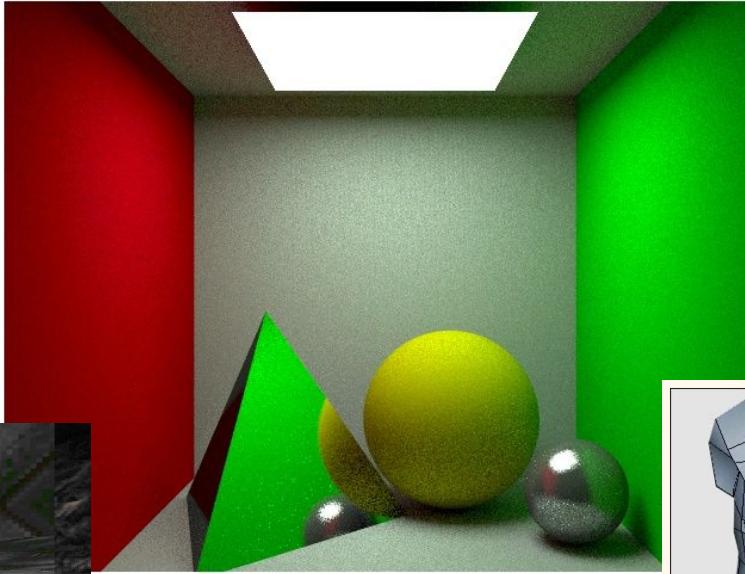
# Refresher for [B]ST

- K-ary (usually binary) trees
- Built upon comparable keys (scalars)
- Similar search procedure
- Preserved balance property, ensures  $O(\log(N))$  max path length
- Can be *homogeneous* (AVL) and not (*B+* tree)



But what if we have vectors?

# Originated from Computer Graphics



1,200 Polys

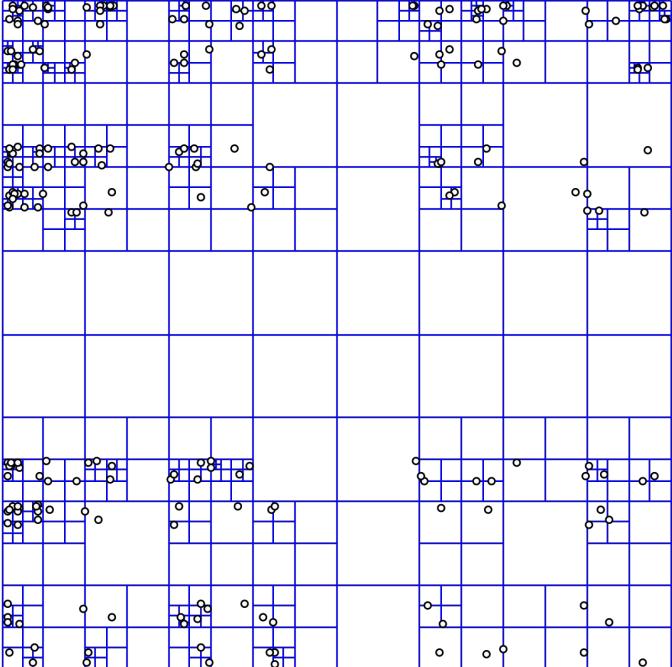
11,600 Polys<sup>6</sup>

# Trivial case: vector is a scalar

- Binary search trees:
  - Splay, RB, AVL trees are best for RAM
- N-ary search trees:
  - B-trees, LSM-trees are used with hard drives
- **Search:**
  - **Exact search is  $O(\log(N))$**
  - **$K$  nearest neighbour search  $O(\log(N) + K)$**
  - **Range search  $O(\log(N) + K)$**

# QuadTree (1974)

- Forms of Quad Trees:
  - Region
  - Point
  - Edge
  - Polygon
- All forms of quadtrees share some common features:
  - decompose space into **adaptable** cells
  - Each cell (or bucket) has a **maximum capacity**.  
When maximum capacity is reached, the bucket **splits**



# QuadTree search

```
function queryRange(range) {  
    pointsInRange = [];  
    if (!this.boundary.intersects(range))  
        return pointsInRange;  
  
    for (int p = 0; p < this.points.size; p++) {  
        if (range.containsPoint(this.points[p]))  
            pointsInRange.append(this.points[p]);  
    }  
    if (this.northWest == null) // no children  
        return pointsInRange;  
  
    pointsInRange.appendArray(this.northWest->queryRange(range));  
    pointsInRange.appendArray(this.northEast->queryRange(range));  
    pointsInRange.appendArray(this.southWest->queryRange(range));  
    pointsInRange.appendArray(this.southEast->queryRange(range));  
    return pointsInRange;  
}
```

# QuadTree insertion #1

```
function insert(p) {  
    if (!this.boundary.containsPoint(p))  
        return false; // object cannot be added  
    if (this.points.size < QT_NODE_CAPACITY && northWest == null) {  
        this.points.append(p);  
        return true;  
    }  
    if (this.northWest == null) this.subdivide();  
  
    if (this.northWest->insert(p)) return true;  
    if (this.northEast->insert(p)) return true;  
    if (this.southWest->insert(p)) return true;  
    if (this.southEast->insert(p)) return true;  
}
```

# QuadTree insertion #2

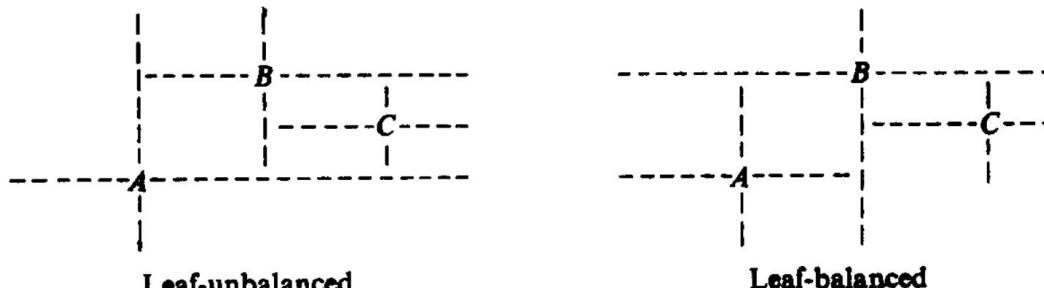


Fig. 2. Single balance

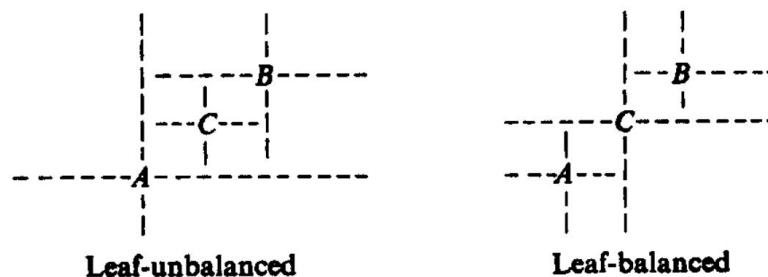


Fig. 3. Double balance

Table 2. Data on leaf-balanced insertion

# QuadTree deletion

<<... In fact, it seems that **one cannot do better than to reinsert all of the stranded nodes, one by one, into the new tree. This answer is not very satisfactory, and it is a matter of some interest whether there exists any merging algorithm that works faster than  $n \log n$ , where  $n$  is the total number of nodes in the two trees to be merged...**>>

# QuadTree optimization

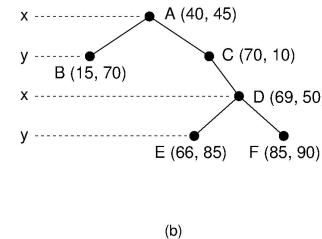
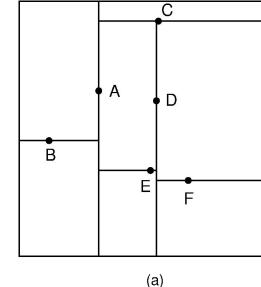
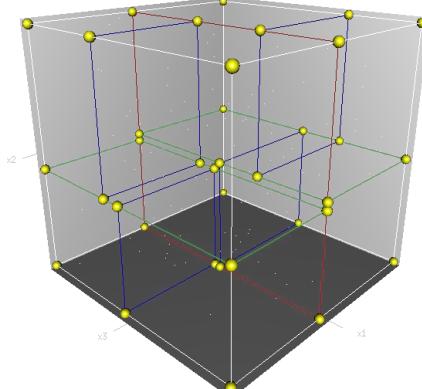
By an **optimized tree** we will mean a quad tree such that every node K has this property: **No subtree of K accounts for more than  $\frac{1}{2}$**  of the nodes in the tree whose root is K.

A simple recursive algorithm to complete optimization is this: Given a collection of **lexicographically ordered records**, we will first find one, R, which is to serve as the root of the collection, and then we will regroup the nodes into 4 subcollections which will be the four subtrees of R. The process will be called recursively on each subcollection... No subtree can possibly contain more than half the total number of nodes

Can you see any suboptimality?

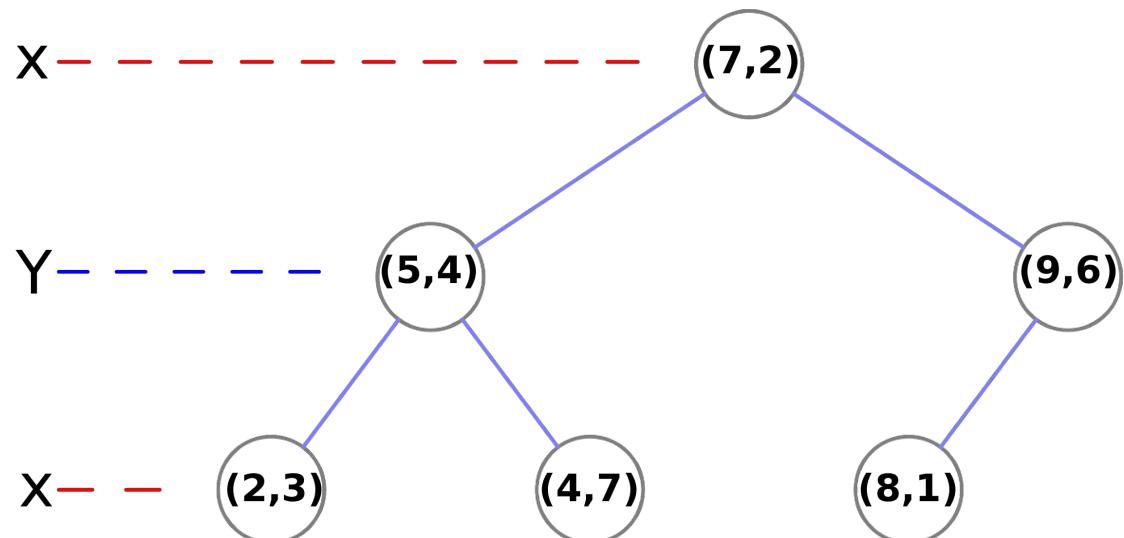
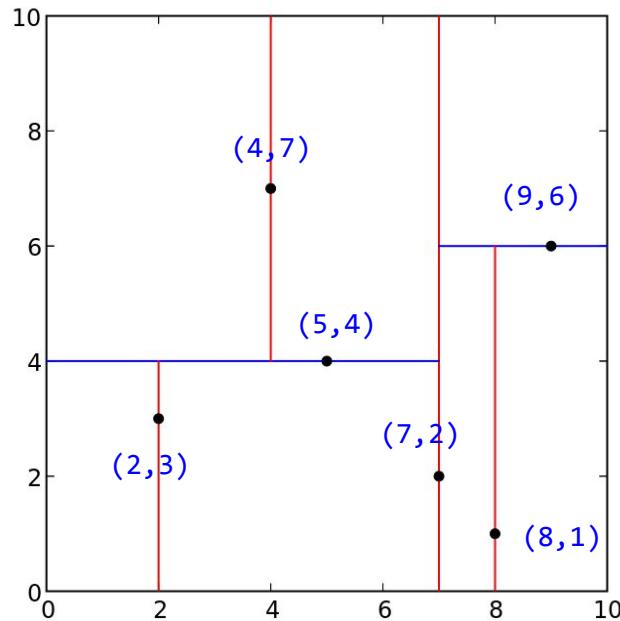
# K-d trees (1975)

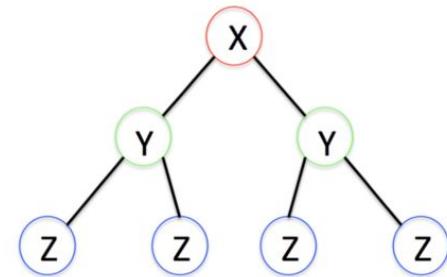
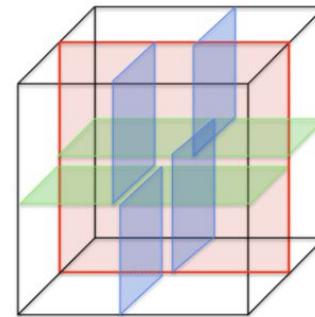
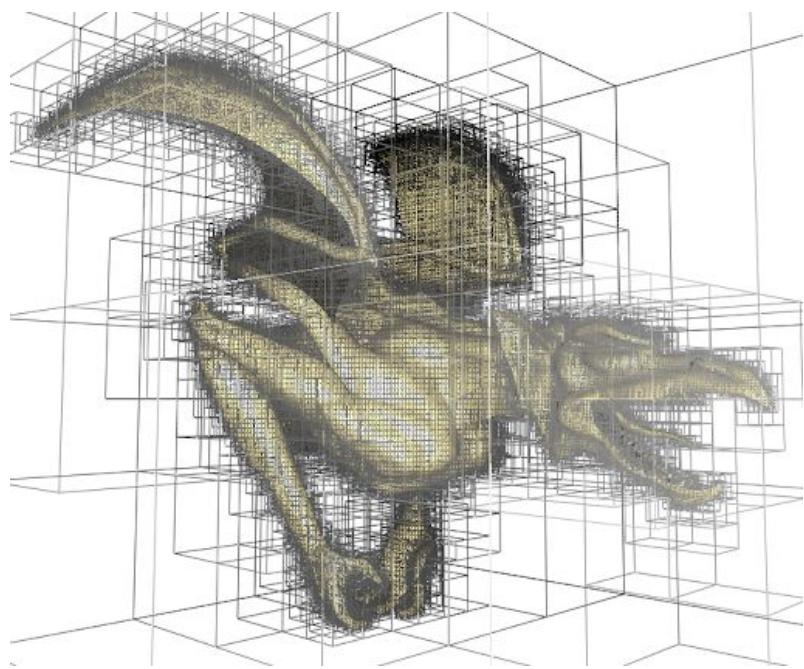
Ideas:



- Split points in 2 **equal** (by #points) subspaces, not 4
- Use **alternating coordinates** at each level  
( $x, y, z, x, y, z, \dots$ )
  - Thus, we need 2 levels to encode quadrants, but they are **equal**
  - And yes, this allows us to have **more than 2 dimensions**
- Cool demo

# K-d trees: building example





# K-d trees

Construction (“homogeneous”):

```
def buildKDTree(vectors, dim=0):
    if not vectors:
        return None      # stop condition, e.g.
    if len(vector) == 1:
        return Node(vector[0])
    vectors.sort(key = lambda x: x[dim])  # or Selection alg for O(N)
    med = len(vectors) // 2                # this will work only for no dups!
    left, med, right = vectors[:med], vectors[med], vectors[med+1:]
    node = Node(med)
    node.left = buildKDTree(left, (dim + 1) % K)
    node.right = buildKDTree(right, (dim + 1) % K)
    return node
```

# K-d trees characteristics

Is built in  $O(n(k + \log(n)))$  time

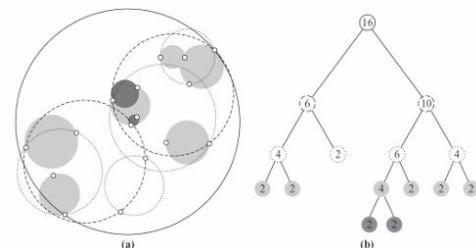
Requires  $O(kn)$  memory (at most node for a point)

Runs range search for  $O(n^{1-\frac{1}{k}} + a)$  where  $a$  — result size

Runs 1-NN search in  $O(\log(n))$  time

To build hyperplanes it requires vector representation of keys

# Ball Trees



Ball Trees and KD-trees are used sklearn implementations of all [nearest neighbour tools](#).

Construction of Ball Tree is almost the same as for KD-tree, with one addition: at each step for pivot element we **compute a radius**.

Thus, we can utilize radius value in kNN search:

**if** `distance(t, B.pivot) - B.radius ≥ distance(t, Q.first)` **then continue;** [[wiki](#)]

*dist to the sphere*                            *dist to the farthest among already selected*

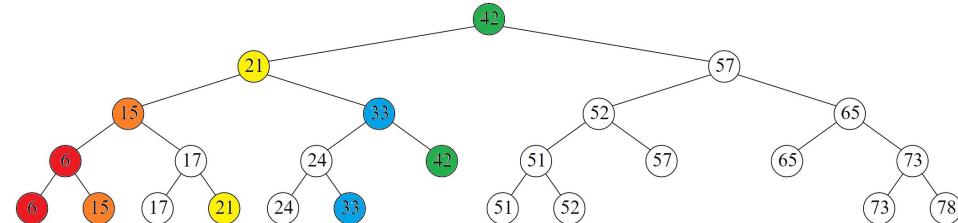
[Here you can read](#) about sophisticated construction algorithms.

*Depending on implementation, you either enumerate dimensions or select the next dimension of the biggest variance*

# Faster range queries - range trees (1979)

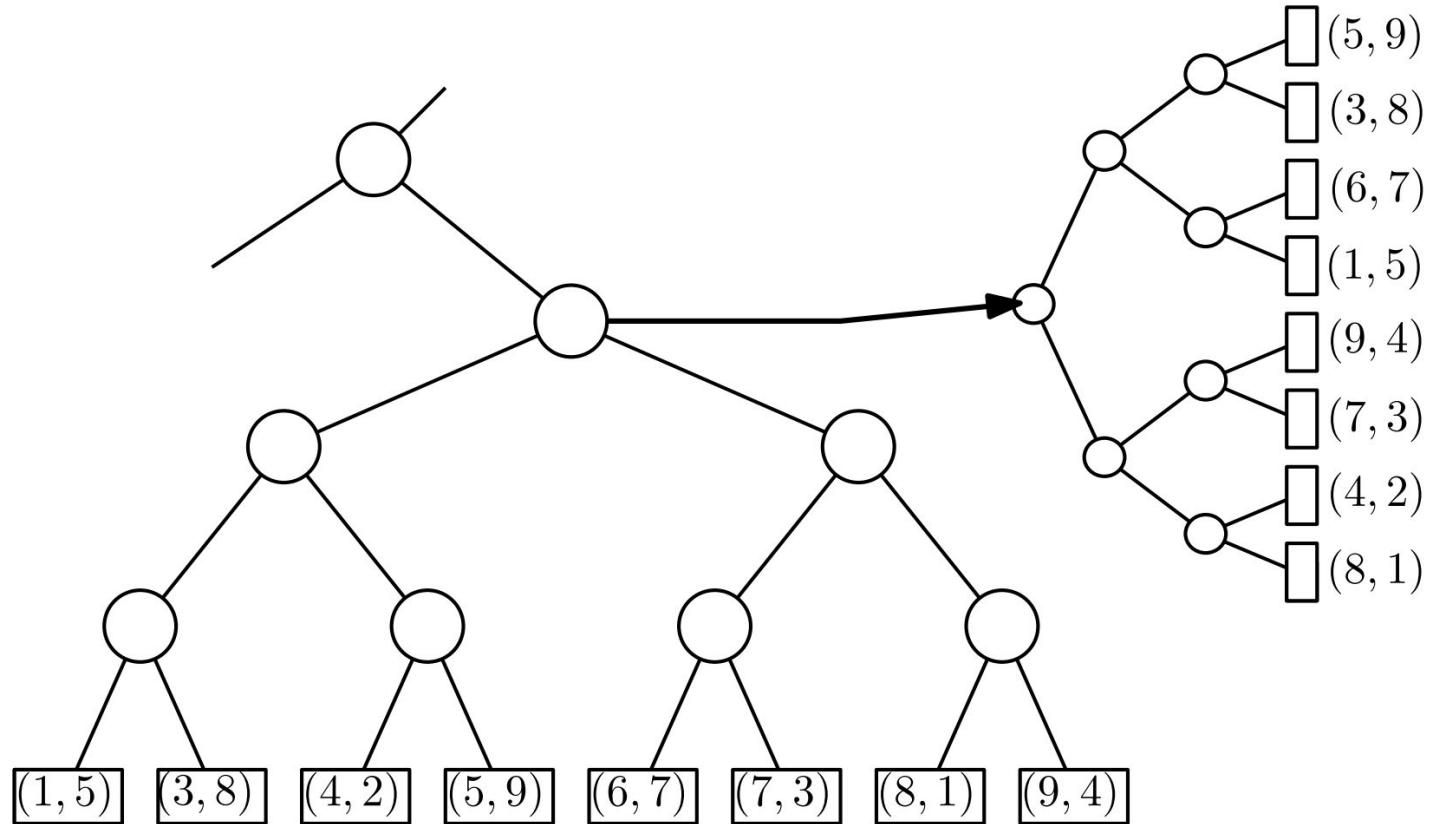
Points are in the leaves.

For 1-dimensional case: balanced non-homogeneous binary search tree on those points. Internal nodes store predecessors (largest to the left)



Range trees in **higher dimensions** are constructed recursively by constructing a balanced binary **search tree on the first coordinate** of the points, and then, for each vertex **v** in this tree, constructing a **(d-1)-dimensional range tree** on the points contained in the **subtree of v**

## Image source link



Sorting and looking for median is soooo  
boring...

## Johnson-Lindenstrauss lemma

... **low-distortion embeddings** of points from high-dimensional into low-dimensional Euclidean space. The lemma states that a set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that **distances between the points are nearly preserved**.  
(Random projections).

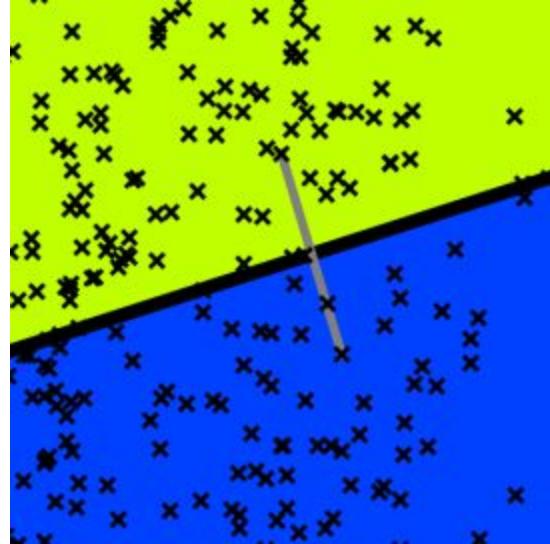
Given  $0 < \varepsilon < 1$ , a set  $X$  of  $m$  points in  $\mathbb{R}^N$ , and a number  $n > 8 \ln(m)/\varepsilon^2$ , there is a linear map  $f : \mathbb{R}^N \rightarrow \mathbb{R}^n$  such that

$$(1 - \varepsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \varepsilon)\|u - v\|^2$$

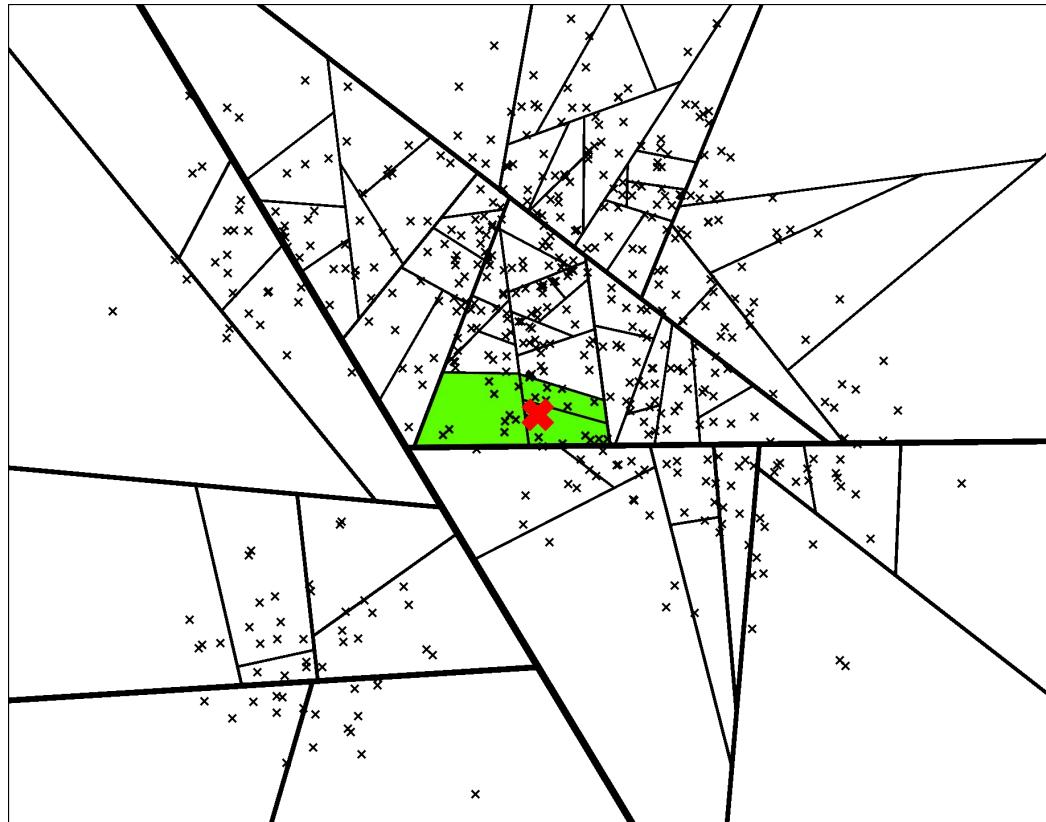
for all  $u, v \in X$ .

# Annoy from Spotify (2015)

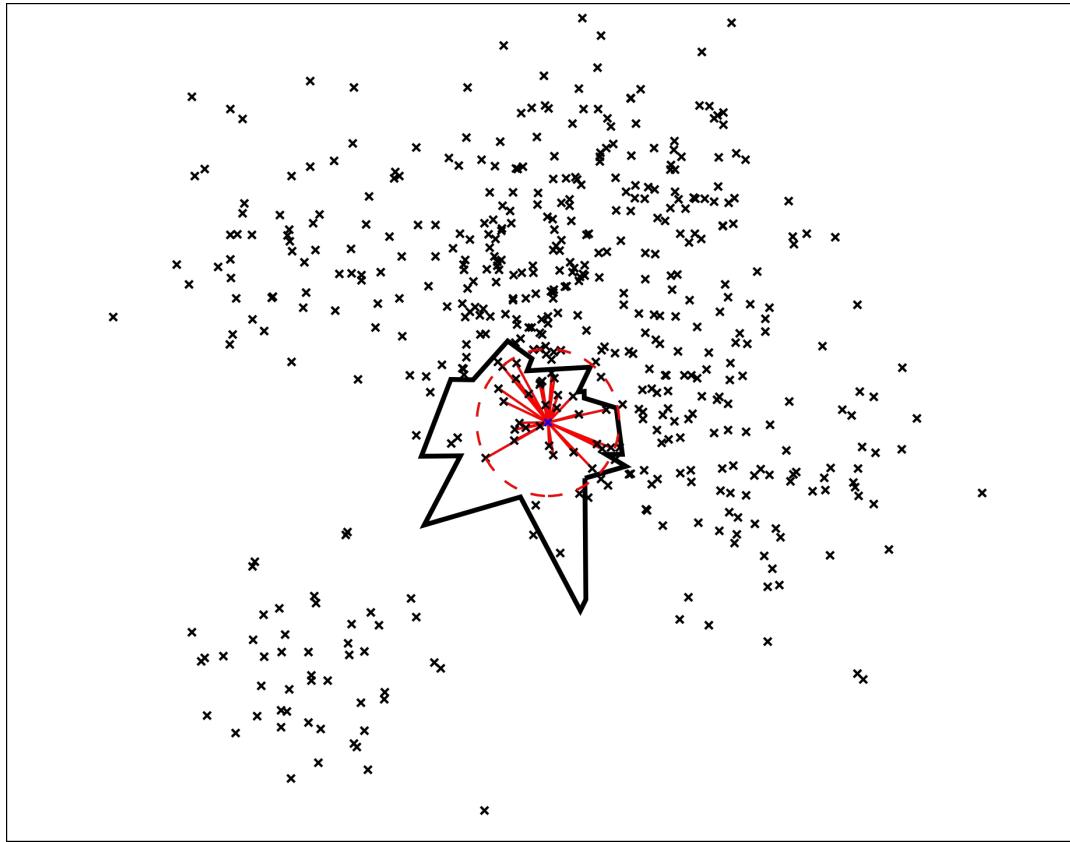
1. Instead of looking for a median, **select equidistant hyperplane for 2 random points** - then split is done in linear time ([random projection](#))
2. Use “**soft threshold**” that allows traversing “wrong” branches for ANNS
3. Build **multiple search trees** over the same dataset (*compare to multiple searches in NSW*)
4. Generalization of binary space partitioning ([BSP-tree](#)) used in CG (Doom, Quake, ...) for visibility sorting.



# Multiple trees ([animation](#))



# ANNS results with Annoy



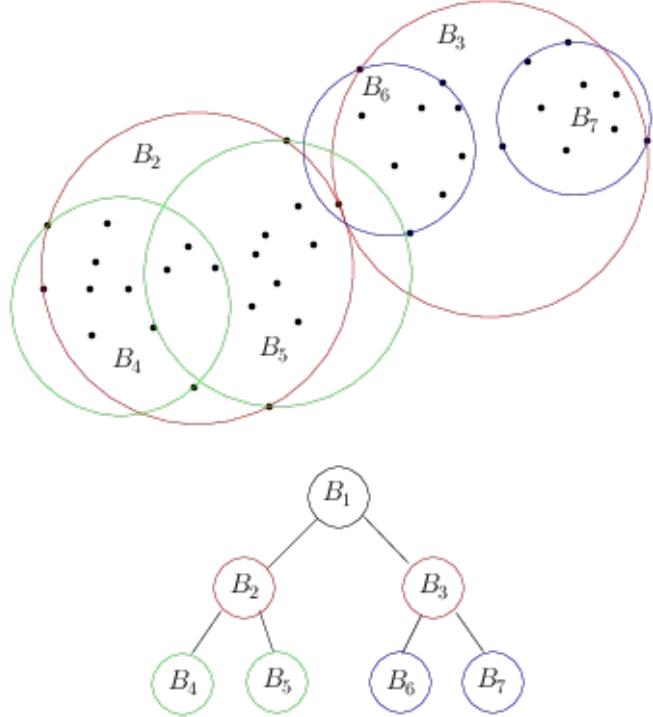
Oh no, I don't have vectors!  
Metric space

# Vantage-point (VP) trees (1991)

Instead of dividing space by a plane, we can divide it by a **sphere** (or nested spheres, recursively). **Sphere** requires only **center** (one of dataset points) and **radius** (which can be estimated in any **metric space**). Radius is selected to split points into equal parts.

```
function Select_vp(S)
    P := Random sample of S;
    best_spread := 0;
    for p ∈ P
        D := Random sample of S;
        mu := Mediand ∈ D d(p, d);
        spread := 2nd-Momentd ∈ D (d(p, d) - mu);
        if spread > best_spread
            best_spread := spread; best_p := p;
    return best_p;
```

```
function Make_vp_tree(S)
    if S = ∅ then return ∅
    new(node);
    node↑.p := Select_vp(S);
    node↑.mu := Medians ∈ S d(p, s);
    L := {s ∈ S - {p} | d(p, s) < mu};
    R := {s ∈ S - {p} | d(p, s) ≥ mu};
    node↑.left := Make_vp_tree(L);
    node↑.right := Make_vp_tree(R)
    return node;
```



SEARCH

# Ok, you must be lost...

All those trees recursively split the space into similar size parts

**Quad Tree** - works in  $R^2$  only. Each node splits space into 4 non equal quadrants.

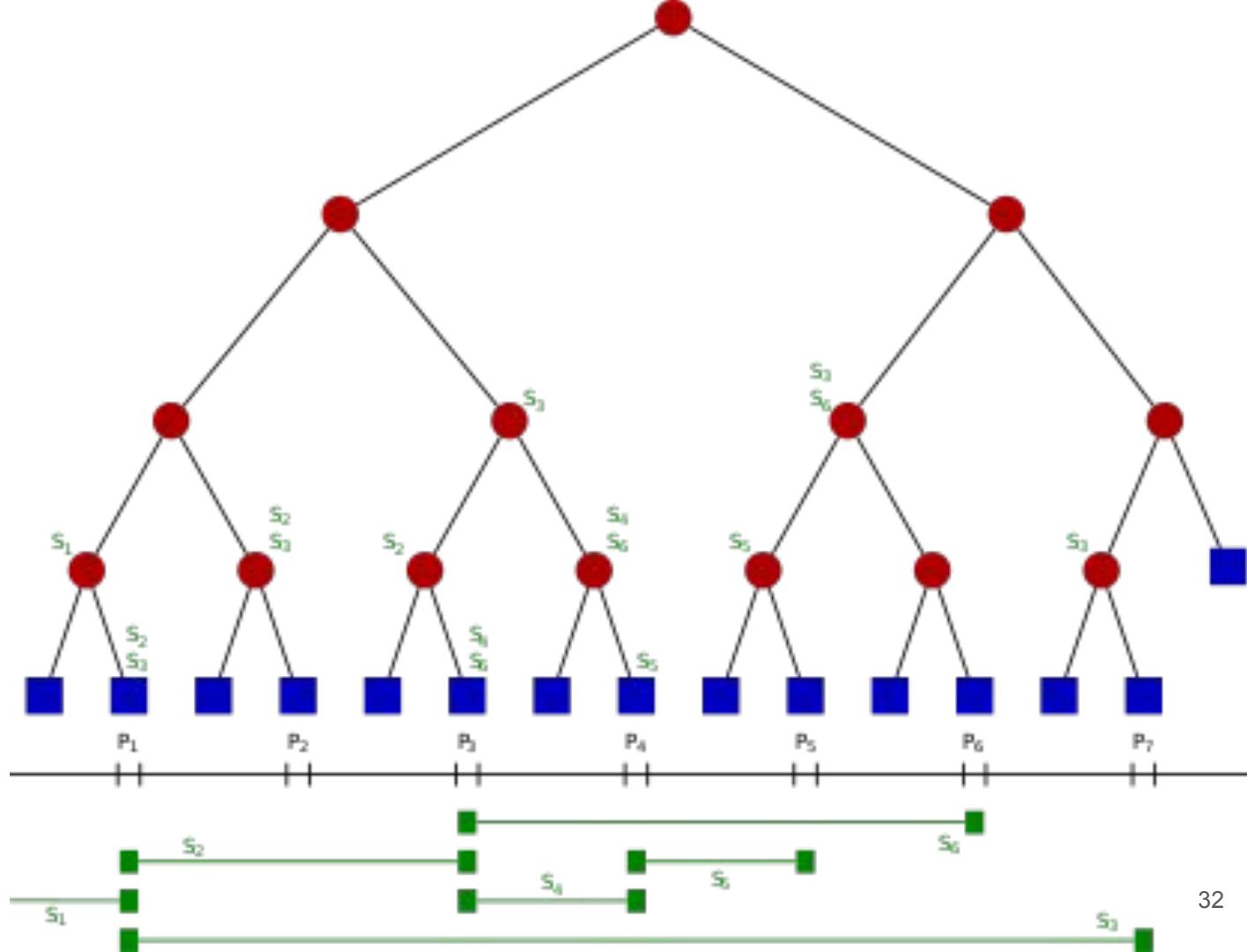
**K-d Tree** - works in  $R^K$ . Each node splits space into 2 equal parts.

**Annoy** - works in  $R^K$ . Instead of sorting and finding median - uses random separating hyperplanes. But compensate with multiple trees

**Vantage-point tree** - works for any metric space. Instead of hyperplanes uses spheres.

Offtopic: interval and BSP trees.  
When object is not a point

## Interval tree



# Interval tree

Tree that **holds intervals** and allows to search fast which of them overlap the query (point or interval).

Construction( $L$ ):

1. You have a list of intervals  $L$ .
2. By  $X_{center}$  split all intervals into “left”, “intersecting”, “right” lists.
3. Store “intersecting” in current node in 2 lists (sorted by start and by end).
4. Run Construction(“left”) and Construction(“right”) intervals.

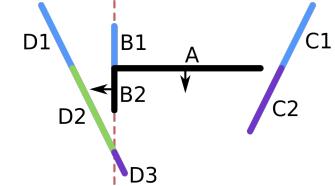
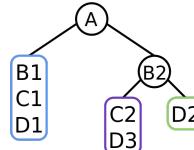
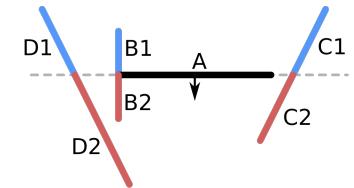
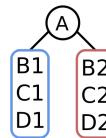
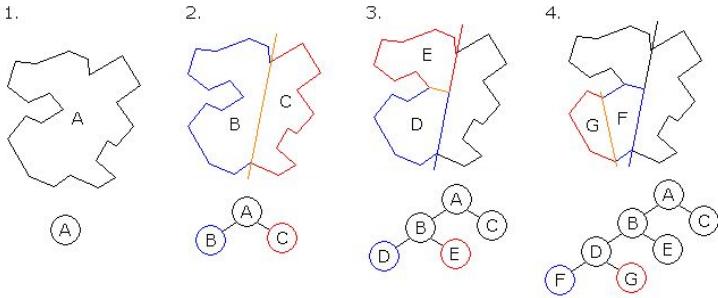
Search( $p$ , node):

1. Compare  $p$  to  $node.X_{center}$
2. Use sorted list in node to find intersecting
3. Go Search( $p$ ,  $node.[left|right]$ ) with respect to [1]

# BSP-tree

To store polygons in a list:

- Choose a polygon  $P$  from a list  $L$ .
- Make a node  $N$ , and add  $P$  to the list of  $N$ .
- For each other polygon  $Q$  in the list:
  - If  $Q$  is in front of  $P$  plane, move  $Q$  to the list  $L_F$  “**in front of  $P$** ”.
  - If  $Q$  is behind  $P$  plane, move  $Q$  to the list  $L_B$  “**behind  $P$** ”.
  - If  $Q$  intersects  $P$  plane, **split** it into two polygons and move them to the respective lists.
  - If that polygon lies in the plane containing  $P$ , add it to the **list of  $N$** .
- Apply this algorithm to  $L_F$  and  $L_B$ .



## See also

[M-trees](#)

[R-trees](#) and [R\\*-trees](#)

[Octree](#)

...

# Relevance feedback and query expansion. Tuning the que\_

Stanislav Protasov

# Agenda

- What if results for the query are *not satisfactory*?
  - *Local* methods of improvement
  - *Global* methods
- How to suggest continuation

*Based on chapter 9*

# Relevance feedback

**Relevance feedback** is using **explicit or implicit user's input** to improve search results. Idea is to use this input as a **navigator in vector space** to drift towards better results.

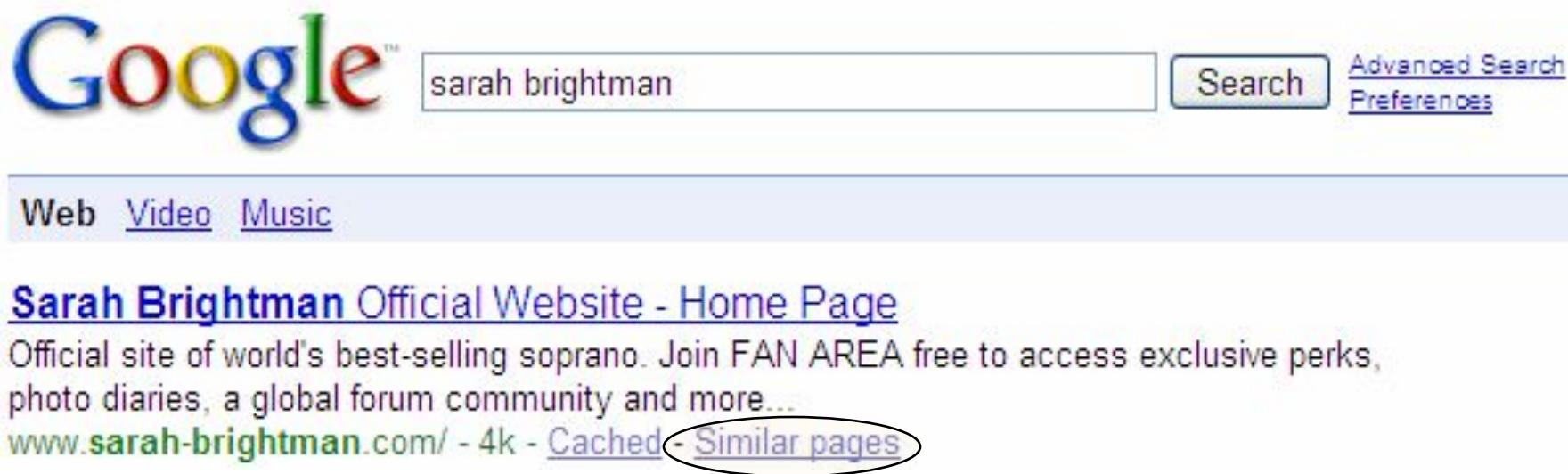
# Relevance feedback

User **feedback** on relevance of docs in initial set of results:

1. User issues a (short, simple) query
2. The **user marks** some results as relevant or non-relevant.
3. The **IR system computes** a better representation of the information need based on feedback.
4. Relevance feedback can go through one or more **iterations**.

Idea: it may be difficult to formulate a good query when you don't know the collection well, so **iterate**

# Similar pages IRL 2009



The image shows a Google search results page. The search bar contains the query "sarah brightman". Below the search bar are three navigation links: "Web", "Video", and "Music". The first result is a link to Sarah Brightman's official website, titled "Sarah Brightman Official Website - Home Page". The description for this result mentions the official site of the world's best-selling soprano, the FAN AREA, photo diaries, a global forum, and more. Below the link, the URL "www.sarah-brightman.com/" is shown, followed by "- 4k - Cached - Similar pages". The "Similar pages" link is highlighted with a red oval.

Google™

sarah brightman

Search Advanced Search Preferences

Web Video Music

[\*\*Sarah Brightman\*\* Official Website - Home Page](#)

Official site of world's best-selling soprano. Join FAN AREA free to access exclusive perks, photo diaries, a global forum community and more...

www.sarah-brightman.com/ - 4k - [Cached](#) - [Similar pages](#)

# Similar pages IRL 2021

The screenshot shows a Google Scholar search results page for the query "hyperplane estimation". The search bar at the top contains "hyperplane estimation". The results section shows three main entries:

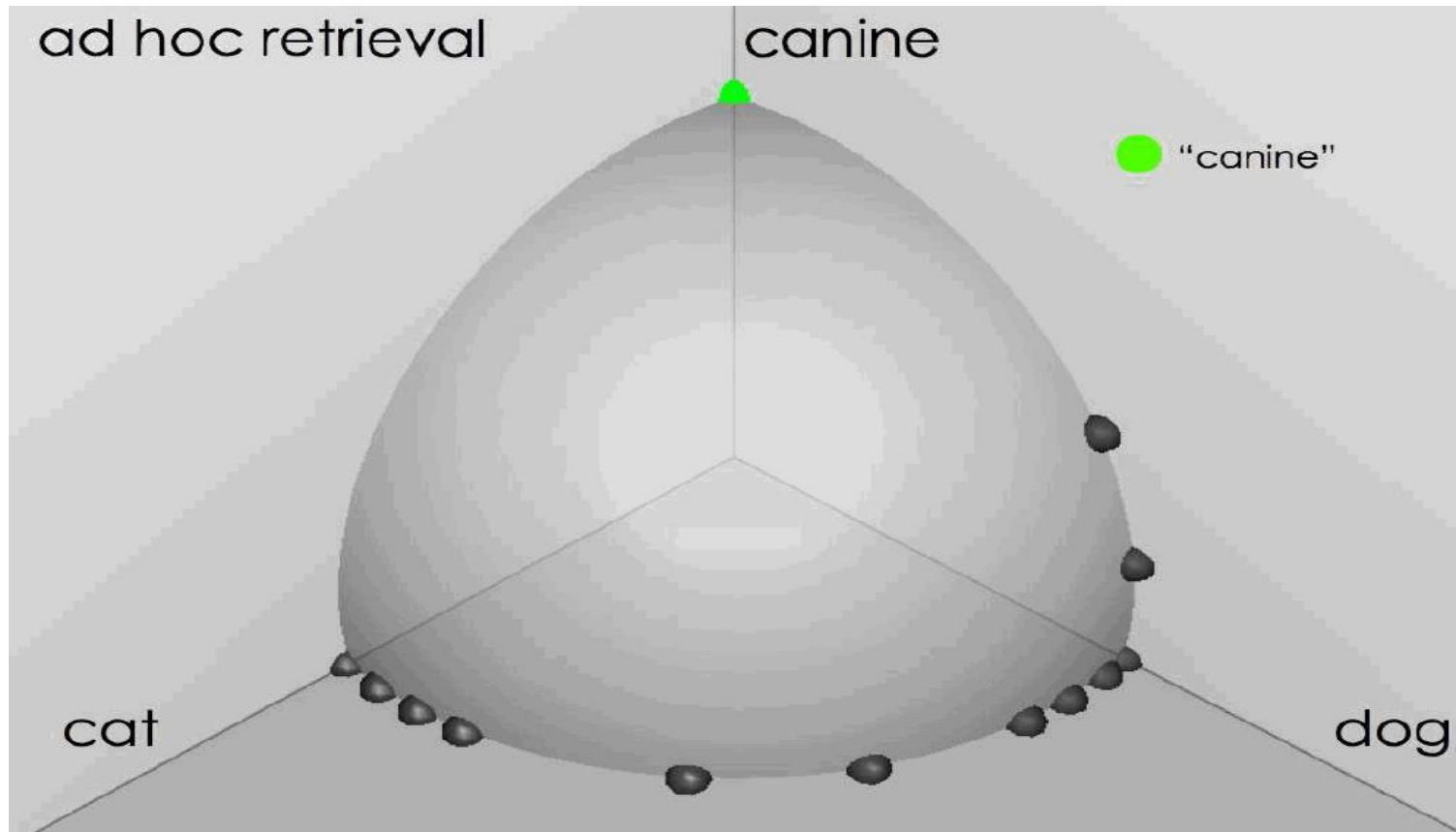
- Hyperplane-based vector quantization for distributed estimation in wireless sensor networks**  
J Fang, H Li - IEEE Transactions on Information Theory, 2009 - ieeexplore.ieee.org  
This paper considers distributed **estimation** of a vector parameter in the presence of zero-mean additive multivariate Gaussian noise in wireless sensor networks. Due to stringent power and bandwidth constraints, vector quantization is performed at each sensor to convert ...  
Cited by 55 Related articles All 5 versions
- Hyperplane method for reachable state estimation for linear time-invariant systems**  
TJ Graettinger, BH Krogh - Journal of optimization theory and applications, 1991 - Springer  
A numerical algorithm is presented for generating inner and outer approximations for the set of reachable states for linear time-invariant systems. The algorithm is based on analytical results characterizing the solutions to a class of optimization problems which determine ...  
Cited by 48 Related articles All 6 versions
- Hyperplane approximation for template matching**  
F Jurie, M Dhome - IEEE Transactions on Pattern Analysis and ..., 2002 - ieeexplore.ieee.org  
... Hager and Belhumeur [6] propose **estimating** this relation by using the inverse of the Jacobian image ... In the case of **hyperplane** approximation, we directly obtain an  $13 \times 42Y \rightarrow 13 \times 41$ . In the [PDF] inria.fr

On the left sidebar, there are filters for "Any time", "Since 2021", "Since 2020", "Since 2017", and "Custom range...". There are also buttons for "Sort by relevance" and "Sort by date". Other options include "include patents" (unchecked), "include citations" (checked), and "Create alert".

The URL in the address bar is [https://scholar.google.com/scholar?q=related:BizW2Wwy4ygJ:scholar.google.com/&scioq=hyperplane+estimation&hl=en&as\\_sd=0](https://scholar.google.com/scholar?q=related:BizW2Wwy4ygJ:scholar.google.com/&scioq=hyperplane+estimation&hl=en&as_sd=0).

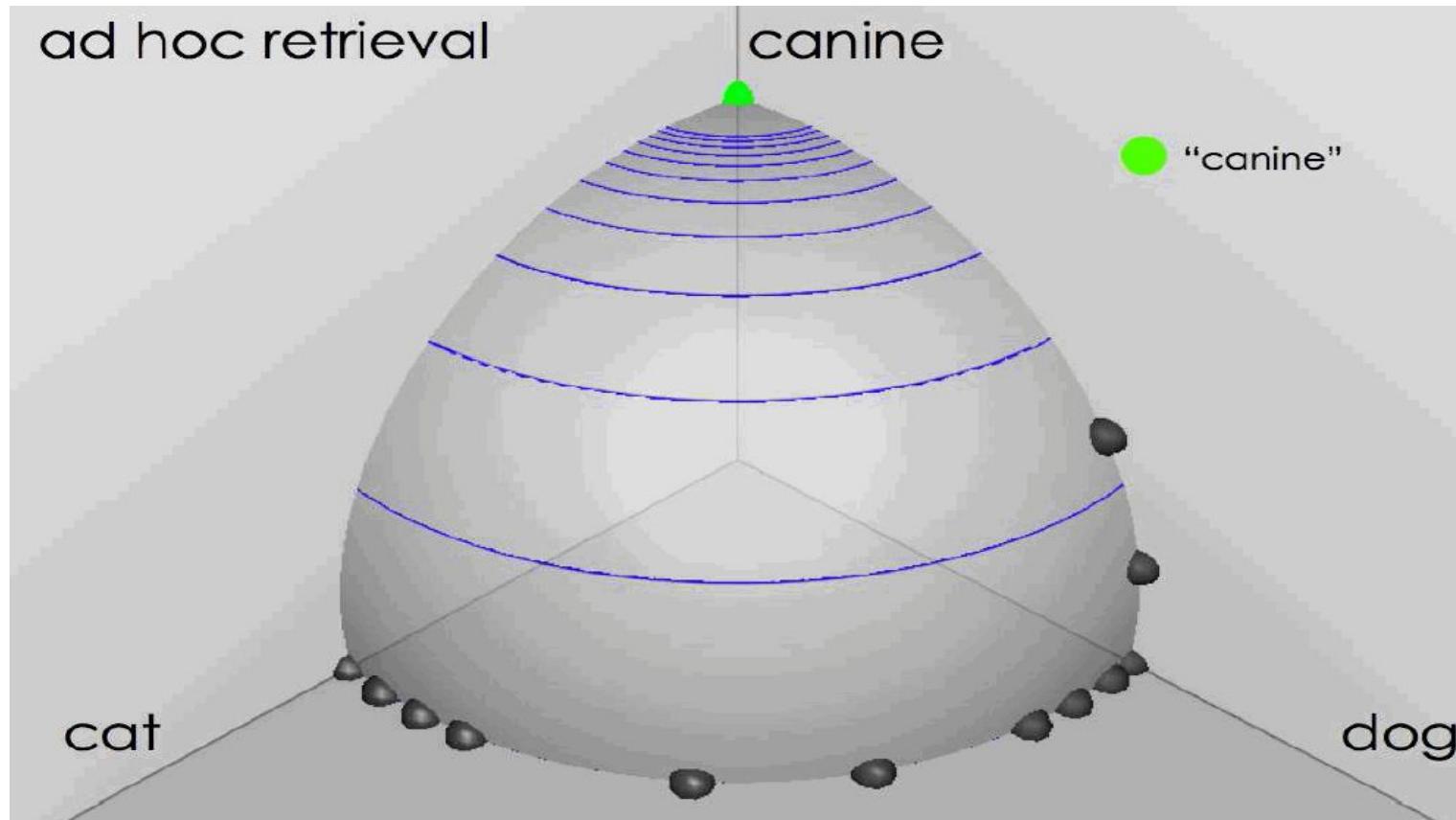
# Ad hoc results for query *canine*

source: Fernando Diaz



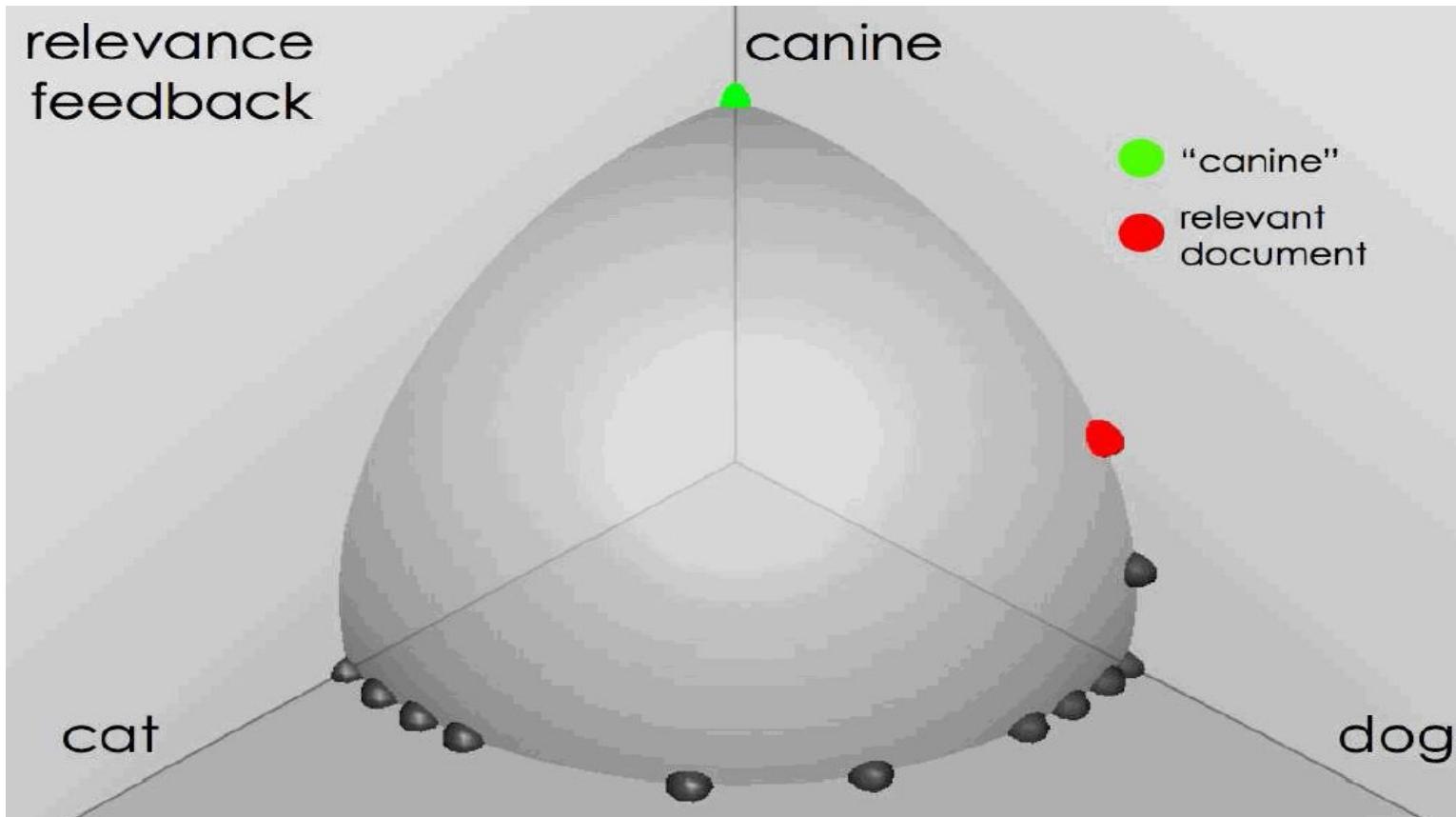
# Ad hoc results for query *canine*

source: Fernando Diaz



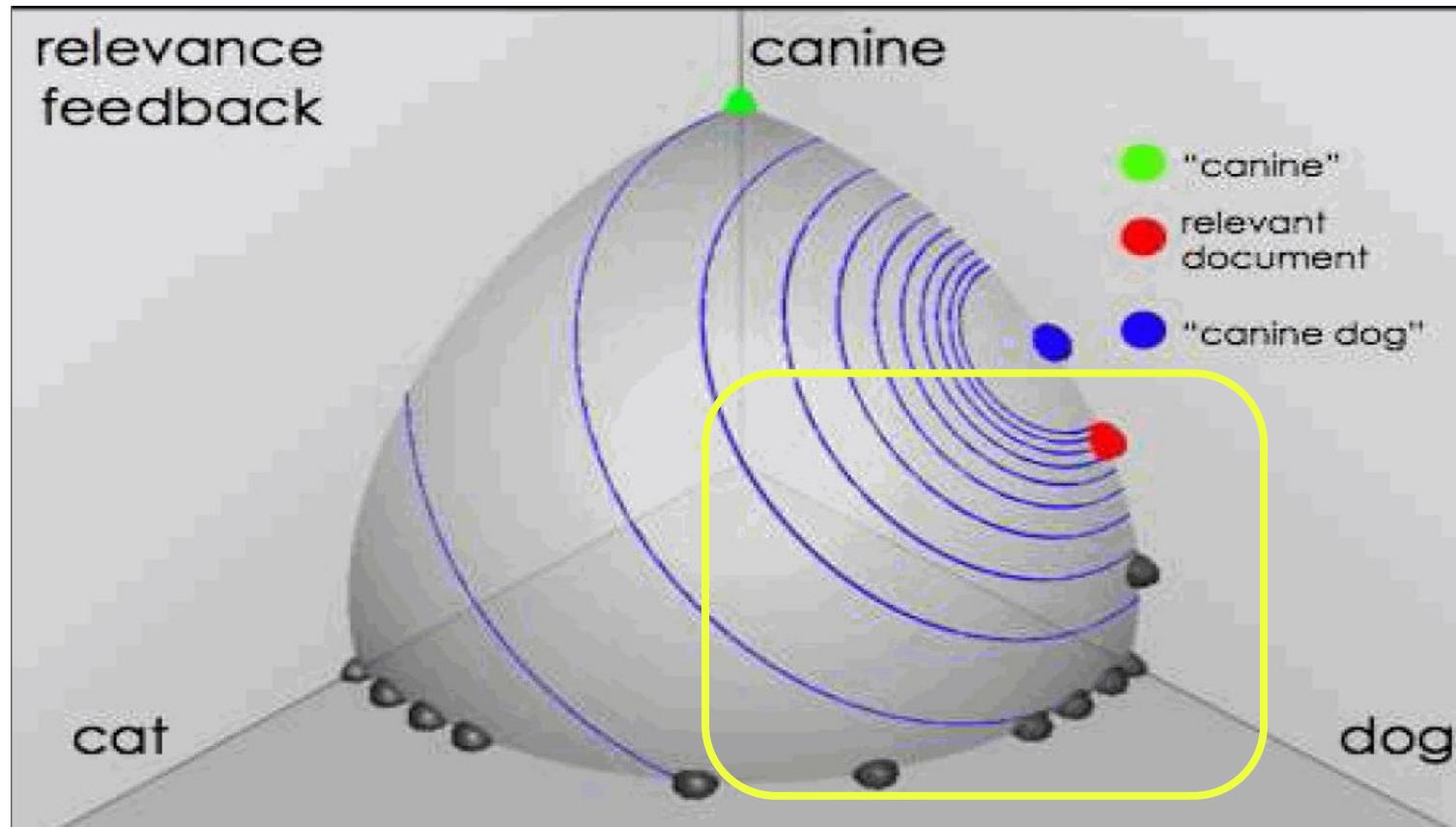
# User feedback: Select what is relevant

source: Fernando Diaz



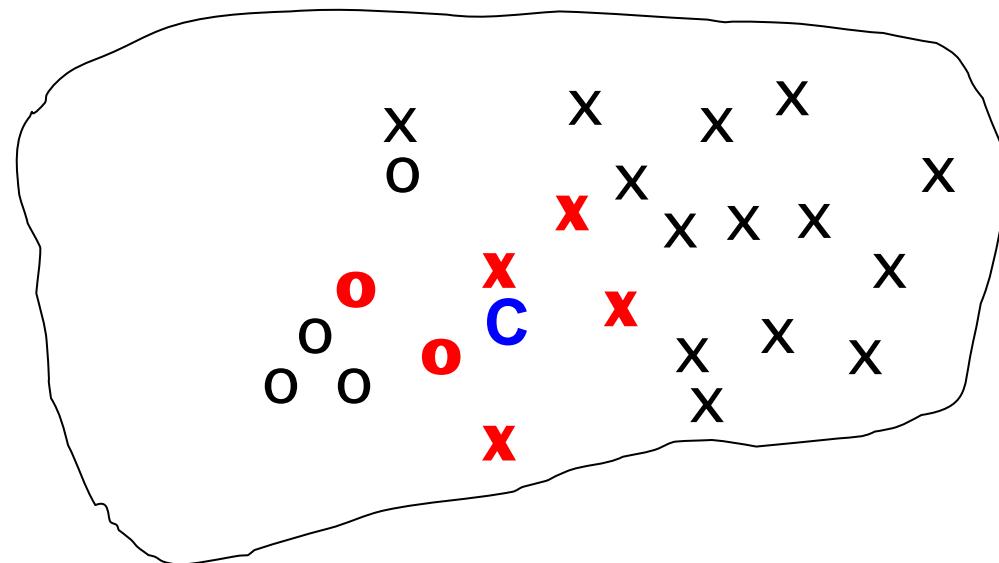
# Results (ranking) after relevance feedback

source: Fernando Diaz

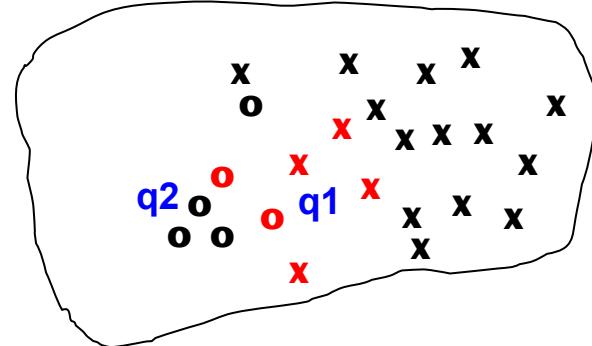


# Key concept: Centroid

- The centroid is the center of mass of a set of points (average vector)



# Relevance feedback idea



- Uses the **vector space model** to pick a relevance feedback query
- Idea: move towards **relevant** and away from **non-relevant**
- Seek the query  $\vec{q}_{opt}$  that maximizes

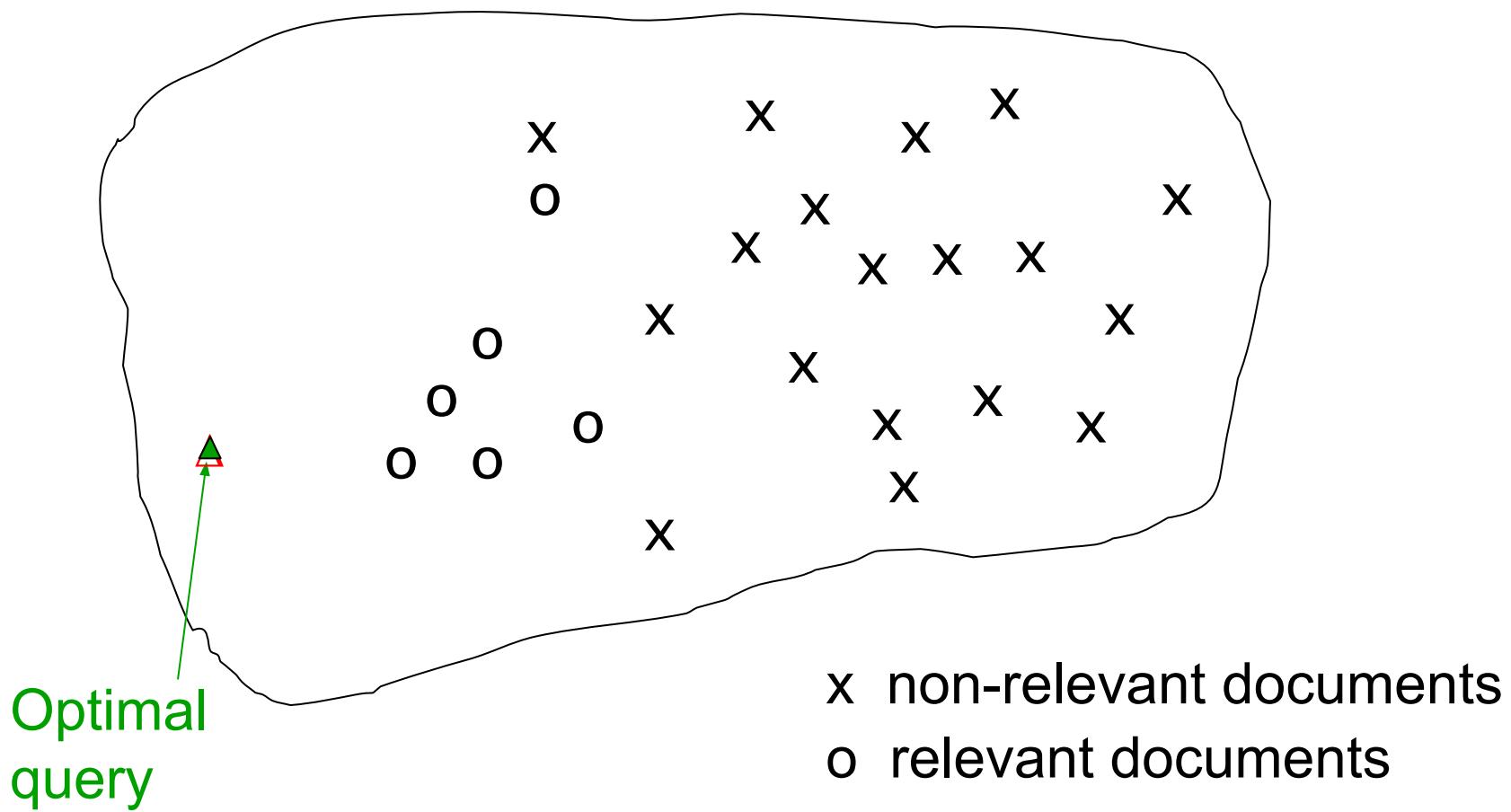
$$\vec{q}_{opt} = \arg \max [\text{sim}(\vec{q}, C_r) - \text{sim}(\vec{q}, C_{nr})]$$

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

How?

Here **C** should be understood as a set of vectors described by a centroid (**D** later in book)

# The Theoretically Best Query



# Problems

1. We don't know **all relevant** documents
2. We excluded **original query** out of consideration ( $q$ )
3. Will it bring us closer to relevant (*average relevant*), or we will jump over and leave a desired cluster (*average irrelevant*)?

# Rocchio explicit algorithm

Kind of **regularization for relevance feedback**, which avoids running away from relevant subspace and original query. Has **recommended parameter values**.

# Rocchio 1971 Algorithm as a framework

- Used in practice:

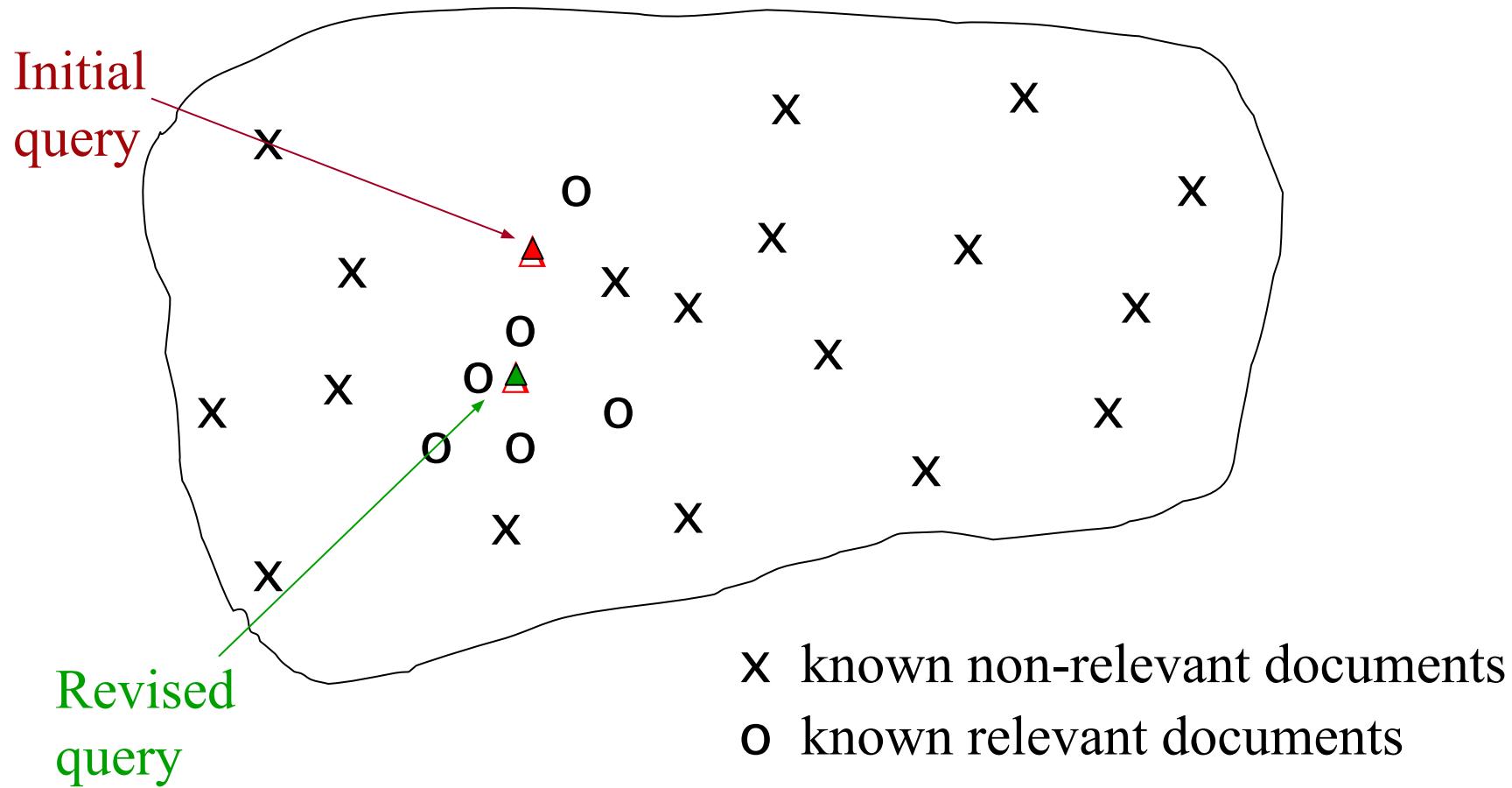
$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

- $D_r$  = set of known relevant doc vectors
- $D_{nr}$  = set of known irrelevant doc vectors
  - Different from  $C_r$  and  $C_{nr}$
- $q_m$  = modified query vector;  $q_0$  = original query vector;  $\alpha, \beta, \gamma$ : weights (hand-chosen or set empirically to 1, .75, .15)
- New query **moves** toward relevant documents and away from irrelevant documents

# Practical comments to framework

- Tradeoff  $\alpha$  vs.  $\beta/\gamma$  : If we have a lot of judged documents, we want a higher  $\beta/\gamma$ .
- We can consider single most similar irrelevant document
- Mostly in practice **improves recall**, not precision

# Relevance feedback on initial query



# Relevance feedback overview

- We can **modify the query** based on relevance **feedback** and apply vector space model.
- Use only the docs that were marked.
- Relevance feedback can **improve recall** and precision, but...
- Relevance feedback is most useful for increasing *recall* in situations where recall is important
  - Users can be expected to review results and to take time to iterate

# Relevance feedback assumptions

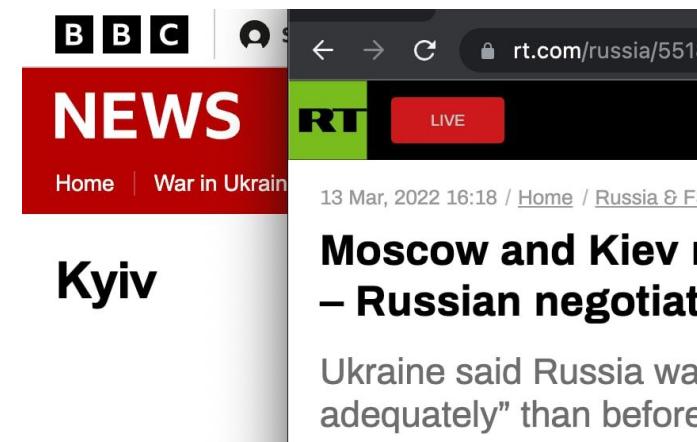
- A1: User has **sufficient knowledge** for initial query.
- A2: Relevance prototypes are “well-behaved”.
  - Term distribution in relevant documents will be similar
  - Term distribution in non-relevant documents will be different from those in relevant documents
    - Either: All **relevant documents are tightly clustered** around a single prototype.
    - Or: There are different prototypes, but they have significant vocabulary overlap.
    - Similarities between relevant and irrelevant documents are small

# Violation of A1

- User does not have sufficient initial knowledge.
- Examples:
  - Misspellings (Brittany Speers).
  - Cross-language information retrieval  
(гиперповерхность).
  - Mismatch of searcher's vocabulary vs. collection vocabulary
    - Cosmonaut/astronaut

# Violation of A2

- There are several relevance prototypes.
- Example:
  - **Pop stars** that worked at **Burger King**
  - Kiev (RT) / Kyiv (BBC)
  - Different vocabularies



# Relevance feedback problems

- Long queries are inefficient for typical IR engine.
- Users are often lazy to provide explicit feedback
- It's often harder to understand why a particular document was retrieved after applying relevance feedback

# Evaluation of relevance feedback

- Use  $q_0$  and compute precision-recall graph
- Use  $q_m$  and compute precision-recall graph
  - Assess on all documents in the collection
    - **Spectacular improvements, but ... it's cheating!**
    - Partly due to **known relevant documents** ranked higher
    - Must evaluate with respect to documents not seen by user
  - Use documents in residual collection (set of documents minus those assessed relevant)
    - Measures usually then **lower than for original query**
    - But a more realistic evaluation
    - Relative performance can be validly compared

# Evaluation of relevance feedback

- Most satisfactory – use two collections each with their own relevance assessments
  - $q_0$  and user feedback from first collection
  - $q_m$  run on second collection and measured
- **Empirically, one round** of relevance feedback is often very useful. Two rounds is sometimes marginally useful.

# Pseudo and implicit feedbacks

User is lazy. Use **top search results** or **user search history** instead of explicit input to improve a query.

# Pseudo relevance feedback

- Pseudo-relevance feedback automates the “**manual**” part of true relevance feedback.
- **Pseudo-relevance algorithm:**
  - Retrieve a ranked list of hits for the user’s query
  - **Assume that the top  $k$**  documents are **relevant**.
  - Do relevance feedback (e.g., Rocchio)
- Works very well **on average**
- But can go horribly wrong for some queries.
- Several iterations will cause query drift.

# Implicit (indirect) relevance feedback

- Ok, we **don't know the actual feedback**
- But we know which documents user or users clicked for other queries
  - For a **single user** consider his/her **preferences** via CTR of the documents through other queries (e.g. “*How to trim a string*” for C++ developer)
  - For overall community select “*relevant*” based on **high CTR**

# Query expansion and suggest

# Query Expansion

- In relevance feedback, users give additional input (relevant/non-relevant) on **documents**, which is used to reweight terms in the documents
- In query expansion, users give additional input (good/bad search term) on **words or phrases**

google.com/search?q=White+stripes&safe=strict&rlz=1C1GCEB\_enRU885RU885&sxsrf=ALeKk03...



White stripes



All

Videos

Images

News

Shopping

More

Settings

Tools

Collections

SafeSearch

jack

album

greatest hits

guitar

rock

stripes seven nation army

de stijl

stripes elephant

the w



The White Stripes – Википедия  
ru.wikipedia.org



Greatest Hits' album  
nme.com



The White Stripes Music and Merchandise  
thirdmanstore.com



The White Stripes - YouTube  
youtube.com

# How do we augment the user query?

- **Manual thesaurus**
  - E.g. MedLine: *physician*, syn: *doc*, *doctor*, *MD*, *medico*
  - Can be *queries* rather than just synonyms
- **Global Analysis:** (static; of all documents in collection)
  - **Automatically derived thesaurus**
    - (co-occurrence statistics)
  - **Refinements based on query log mining**
    - Common on the web
- **Local Analysis:** (dynamic)
  - Analysis of documents in **result set**

# Thesaurus-based auto query expansion

- For each term  $t$  in a query, expand the query with **synonyms** and **related words** of  $t$  from the thesaurus, maybe weighted
  - feline → feline +cat
- Generally increases *recall*
- Widely used in many science/engineering fields
- May significantly *decrease precision*, particularly with ambiguous terms.
  - “**interest rate**” → “interest rate +**fascinate** +**evaluate**”
- There is a high cost of manually producing a thesaurus
  - And for updating it for scientific changes

# Automatic Thesaurus Generation

- Attempt to generate a thesaurus automatically by analyzing the collection of documents
- Fundamental notion: similarity between two words
- *Definition 1: Two words are similar if they co-occur with similar words.*
- *Definition 2: Two words are similar if they occur in a given grammatical relation with the same words.*
  - **Co-occurrence** based is more **robust**,
  - **grammatical relations** are more **accurate**.

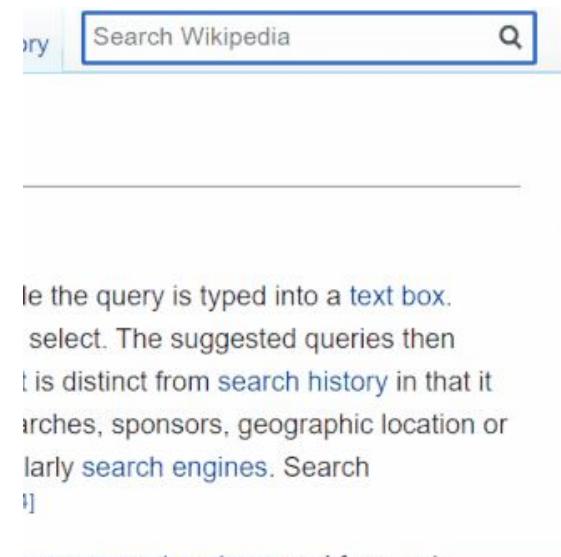
# Automatic Thesaurus Generation Discussion

- **Quality of associations** is usually a problem.
- Term **ambiguity** may introduce irrelevant statistically correlated terms.
  - “Apple computer” → “Apple +red +fruit computer”
- Since terms are **highly correlated** anyway, expansion may **not retrieve** many **additional** documents.

# Suggest

... query feature used in computing to show the **searcher shortcuts**, while the query is typed into a text box. Before the query is complete, a drop-down list with the **suggested completions** appears to provide options to select [[wiki](#)]

- **Blacklist** of what can be a “bad” suggest
- **Complaints** on certain suggestions (bots, law violations, insults)
- **Trie** is the most useful data structure to implement suggestions



If the query is typed into a [text box](#),  
[select](#). The suggested queries then  
[t](#) is distinct from [search history](#) in that it  
[archives](#), [sponsors](#), [geographic location](#) or  
[larly search engines](#). [Search](#)  
[ ]

[management systems](#) and frequent

# Suggest

The screenshot shows a browser window with a search bar at the top. The search term 'Summ' has been typed, and a list of suggestions is displayed below it. The suggestions are:

- G Summ
- Summ - Поиск Google
- summer
- summertime sadness
- summertime
- summary
- summertime sadness текст
- Summer'20 Research Internship Project Submission Form - Googl... - docs.google.com/spreadsheets/d/1-223547NtQVMKBINlkag6uO...
- Project MUSE - A Selection of New Irish Poets - muse.jhu.edu/login?auth=0&type=summary&url=/journals/eire-ireland/v040/40.3fuh...

Thanks for your attention!

# Sound and speech retrieval

Stanislav Protasov

# Agenda

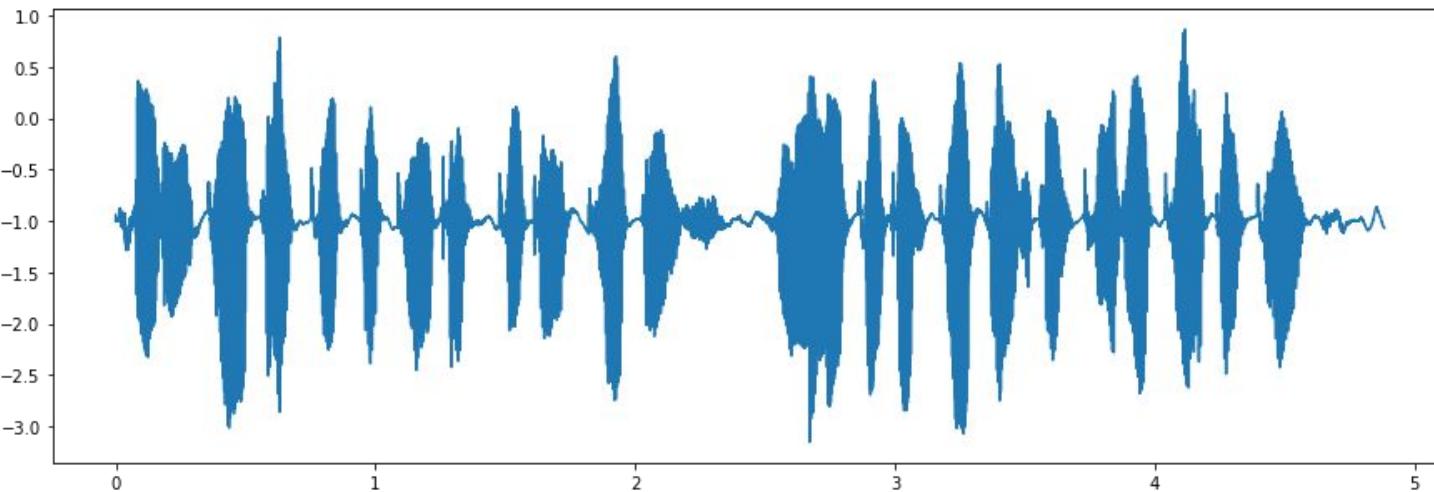
- Sound as a wave
- Music search
- Speech recognition

What is sound and  
how humans perceive it?

***Hint:*** frequencies

# What is the sound?

Sound is a **vibration** that propagates through a transmission medium such as a gas, liquid or solid.



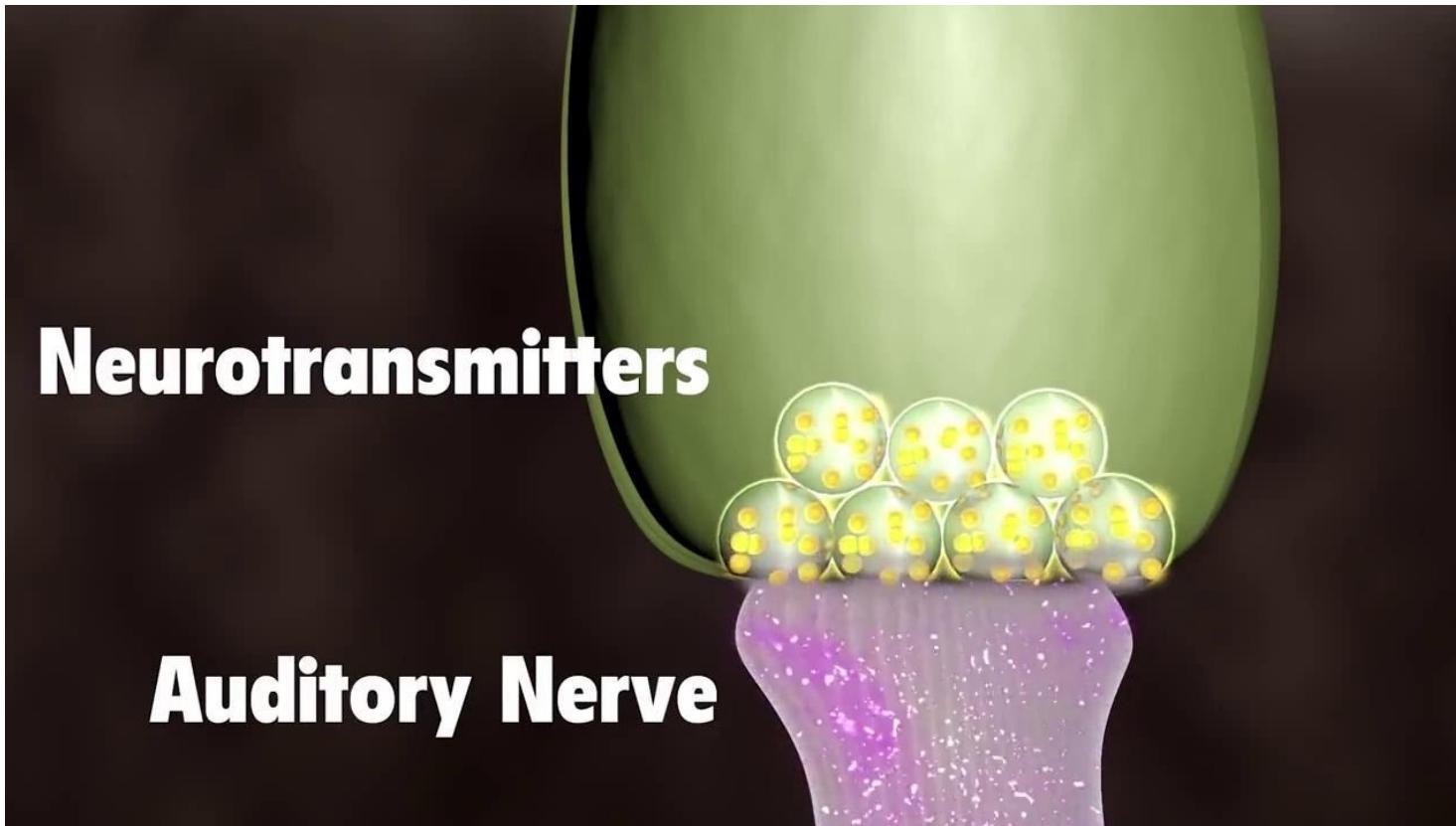
# Vinyl player





National Institute on  
Deafness and Other  
Communication Disorders

# How ear works



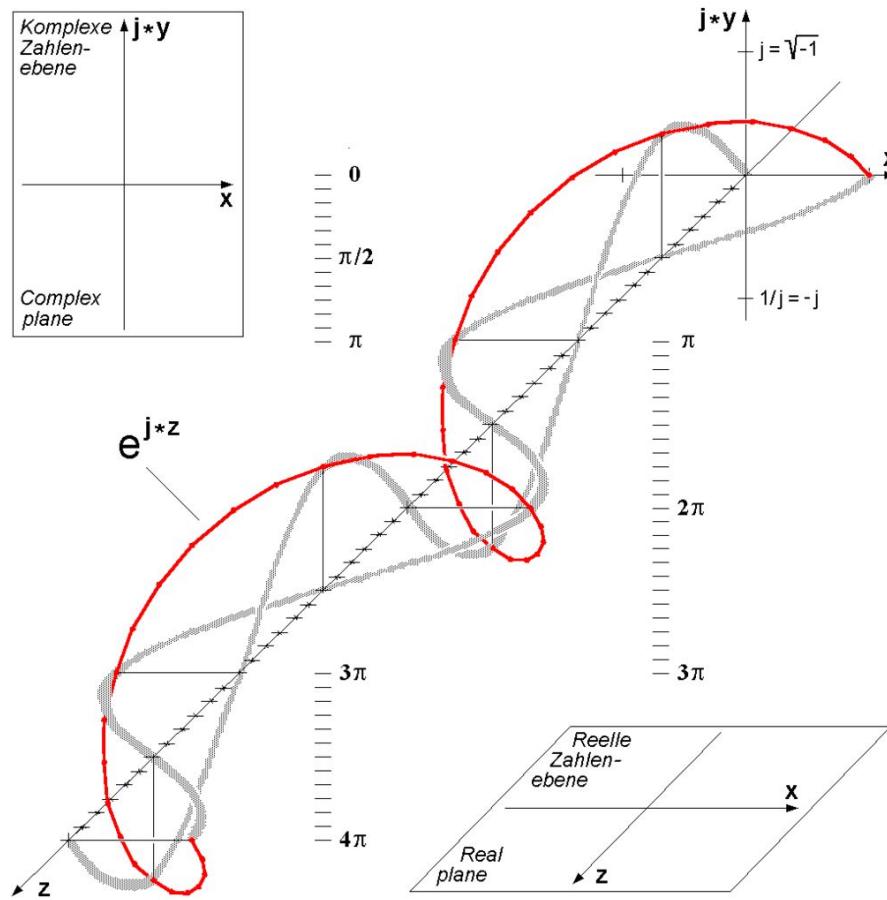
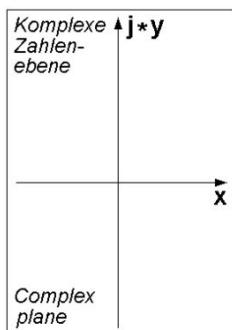
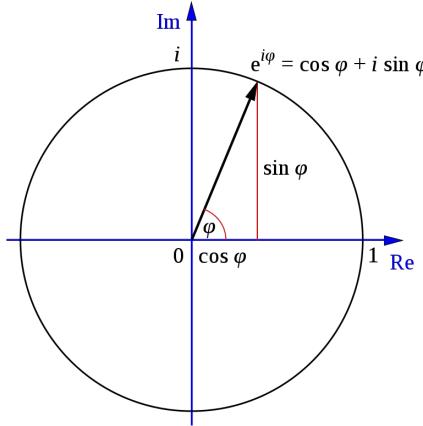


# How to repeat this in math?

*Hint:*  $FT$

# Euler's identity to link complex exponent with frequencies

$$e^{ix} = \cos x + i \sin x,$$



# Fourier Transforms

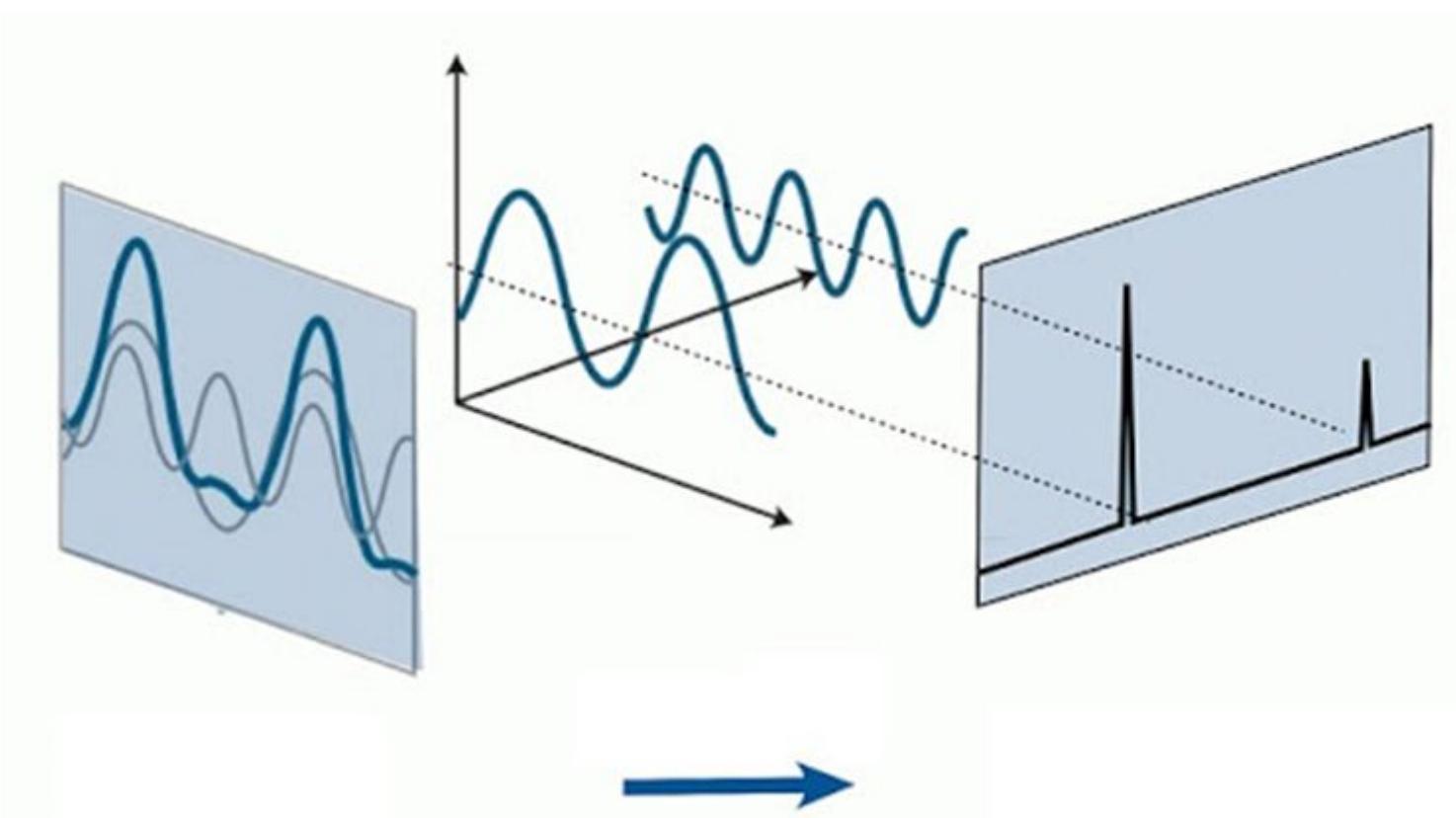
$$\text{FT: } \hat{f}(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i x \omega} dx$$

$$\text{DTFT: } X_T(\omega) = \sum_{n=-\infty}^{+\infty} f(nT) e^{-2\pi i \omega n T}$$

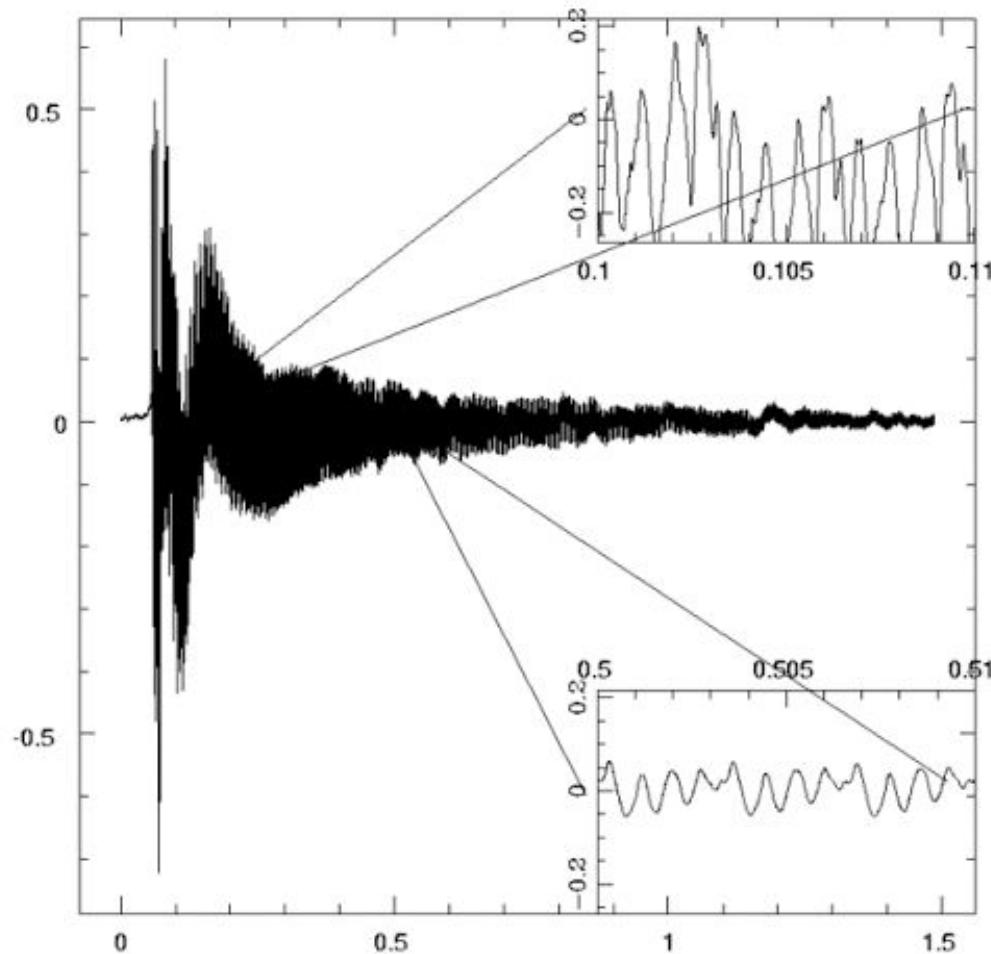
$$\text{DTFT+window: } X_T(\omega) = \sum_{n=0}^M f(nT) w\left(\frac{n}{M}\right) e^{-2\pi i \omega n T}$$

$$\begin{aligned} \text{DFT: } X_{T_N}(k) &= X_T\left(\frac{k}{NT}\right) = && k=0, 1, \dots, N-1 \\ &= \sum_{n=0}^M f(nT) w\left(\frac{n}{M}\right) e^{-2\pi i \frac{kn}{N}} \end{aligned}$$

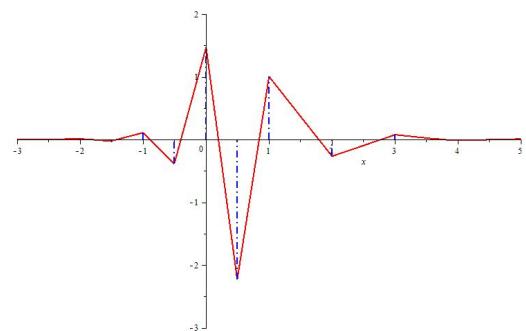
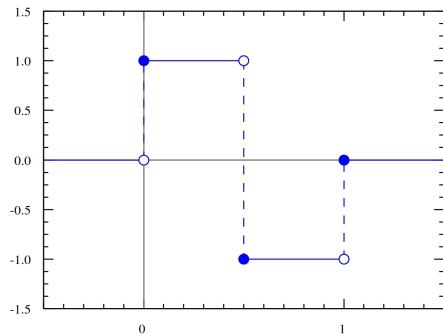
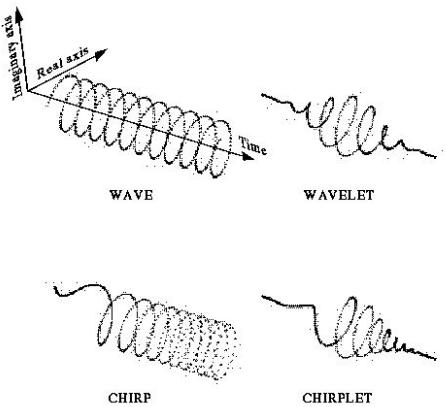
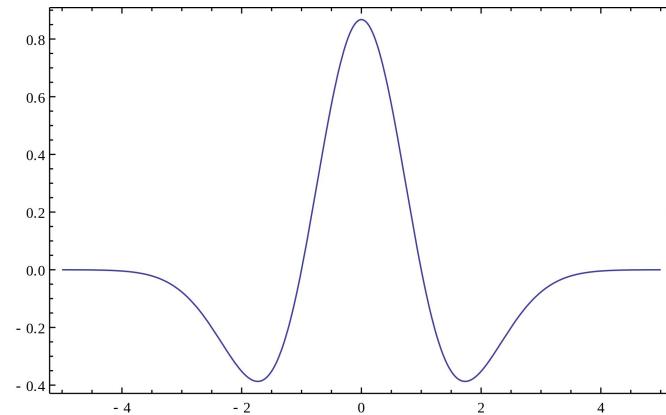
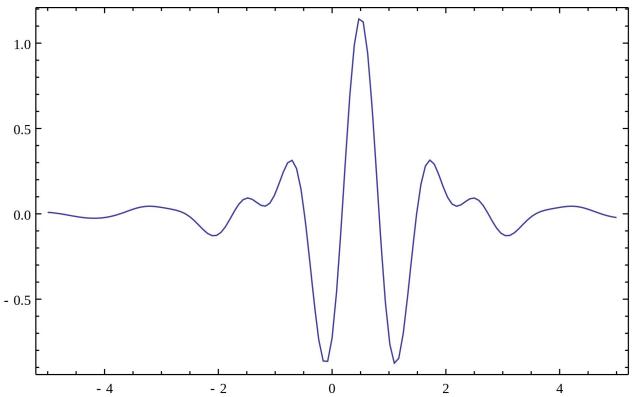
# Fourier transform



# Guitar pitch

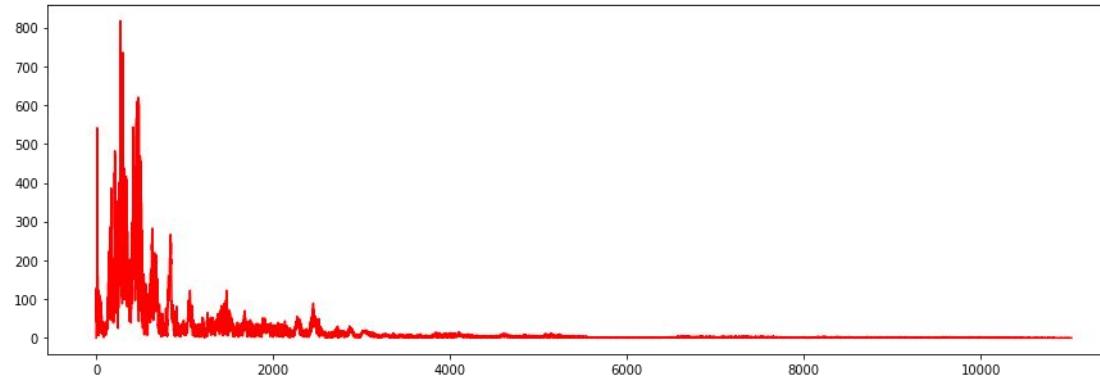


# Wavelets



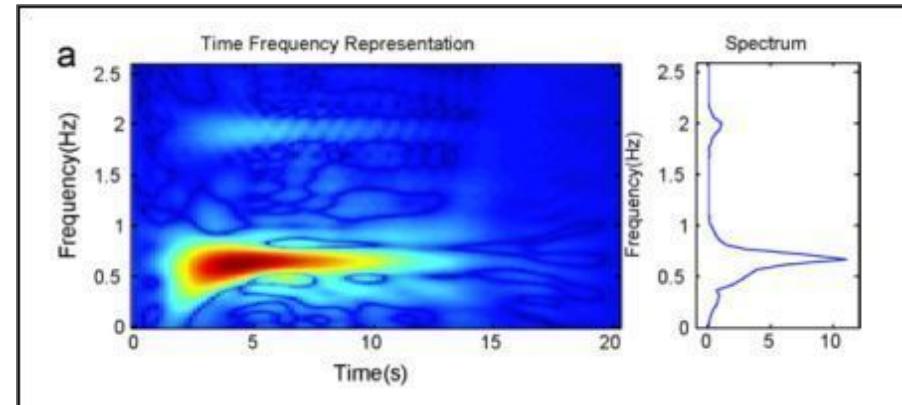
# What is the sound for human?

We percept sound using **frequency** receptors. Each moment looks like this:



Also important — we perceive sounds in **log scale**

Timeline is like this:

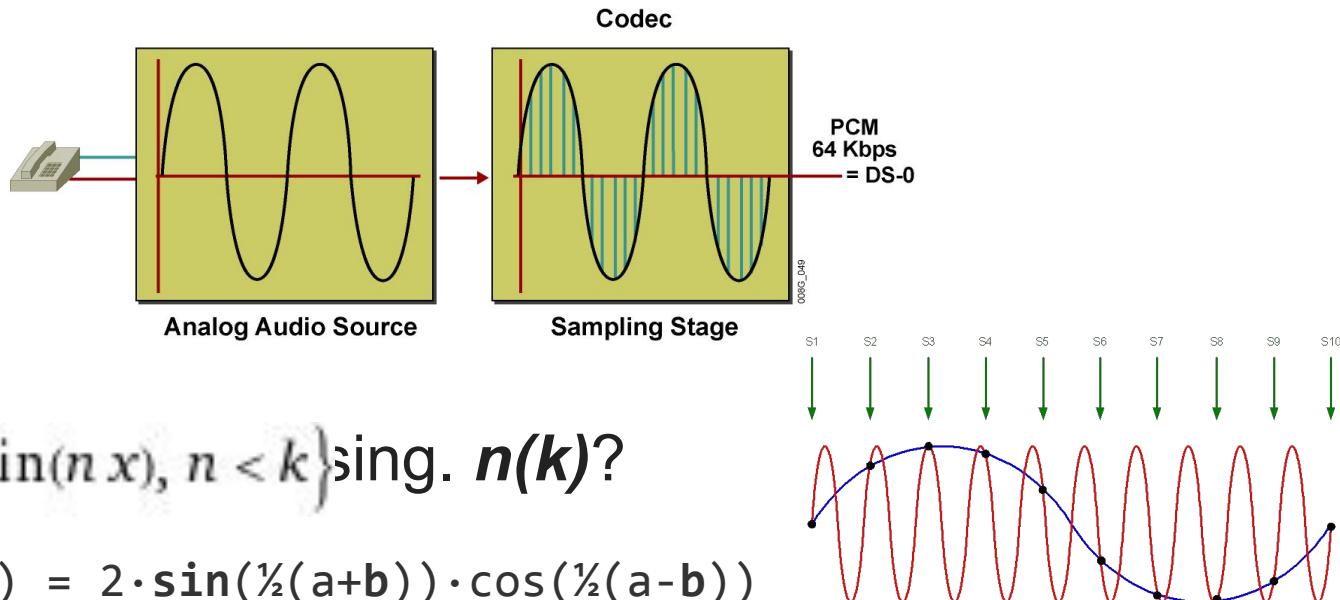


# *Sound recording and playback*

- Digital uncompressed sound consists of regular measurements of signal.
- Measurement frequency is managed using RATE parameter
  - 22050 means 22050 measurements per second  
**(discretization)**
- How accurate we measure in managed is tuned with format  
**(quantization)**
  - How many different amplitude values can be encoded
- Channels — number of inputs/outputs (stereo=2, mono=1)
- $BPS = RATE * CHANNELS * FORMAT$
- Together this is **PCM — pulse code modulation**

# Nyquist-Shannon (Kotelnikov) theorem

If a function  $x(t)$  contains **no frequencies** higher than  $B$  hertz, it is **completely determined** by giving its values at a series of points spaced  $1/(2B)$  seconds apart.



## **Takeaway:**

Ok, machine can represent sound wave  
in human-like form with no information  
loss

# Music fingerprinting

## **Takeaway for exact search:**

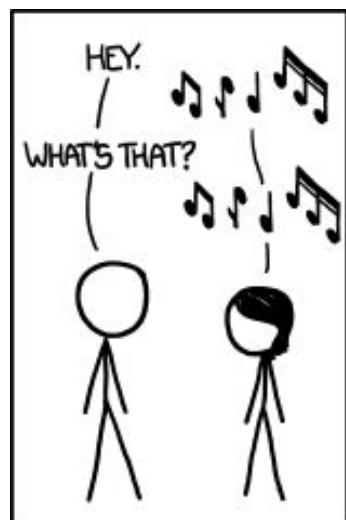
1. Find peaks on spectrogram
2. Use their relative positions in freq-time space as descriptors
3. Maximize descriptors intersection for the query and candidates

# Why?

- I like this song, I want to buy it
- Forensic (when was this song playing)
- **Copyrights** (see youtube or instagram policy)

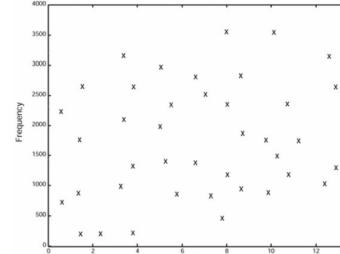
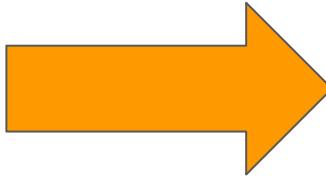
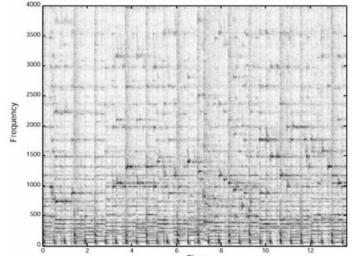
# How to form a query

1. Exact sample
2. Humming



# Shazam exact match algorithm (1)

1. Robust spectrogram. (Log-scale bins of frequencies)



2. Build pairs for hashing (32bit): anchor point + other point from target zone.

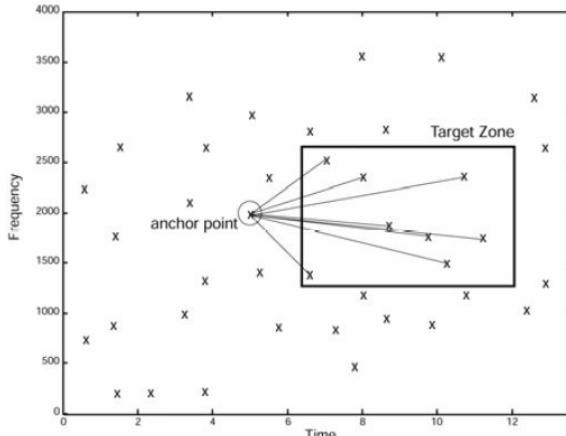


Fig. 1C - Combinatorial Hash Generation

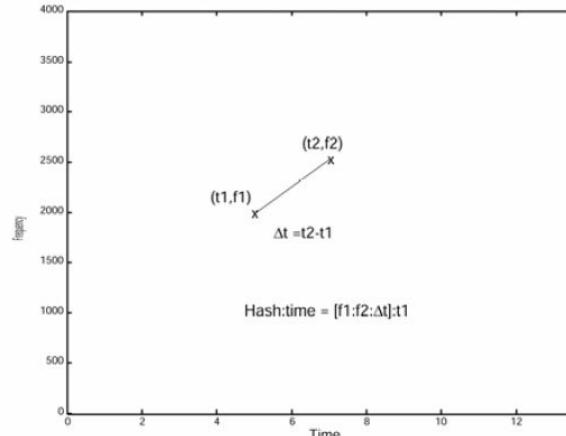


Fig. 1D - Hash details

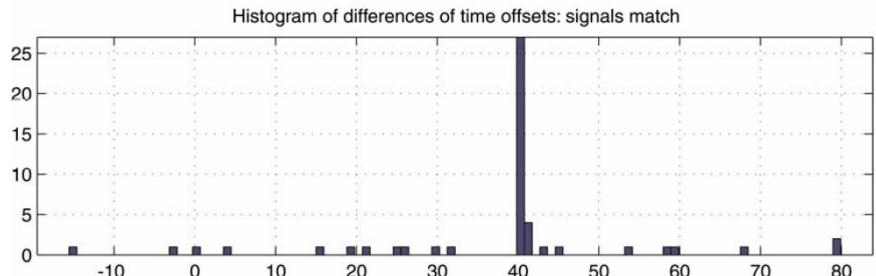
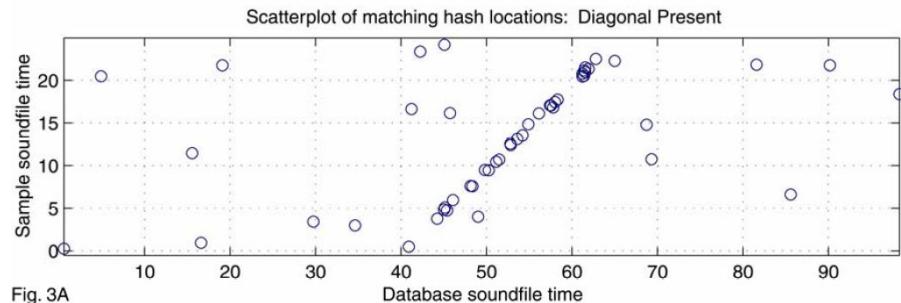
# Shazam algorithm (2)

3. Put those points to a **hashmap**.

Memory:  $[4B \text{ (hash)} + 4B \text{ (val)}] * \text{peaks}$ .

4. Query for songs with a processed sample.

5. Plot a histogram of offsets  
(get times from HT, put  
them in bins), identify  
real offset



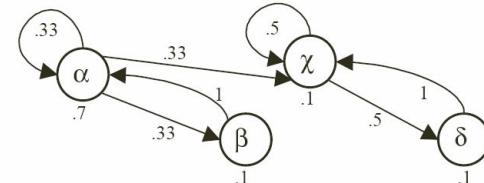
# Other fingerprinting approaches

Exact frequency and tempo are not always important:

- Zero-crossing rate
- Spectrum
- Envelope (spectral flatness, frequency band)
- ...

# Query by Humming (QbH)

- Detect **coarse melodic contour**, retrieve by string search
  - S=same note, U=up, D=down
  - E.g., Beethoven's 5th: – **S S D U S S D**
  - OR U/D/S – but with five contour levels
- Add **rhythm information**
- Use **beat information**
- Use **HMMs** to represent song database
- ***Dynamic Time Warping* (DTW)** based algorithm,  
match waveform directly



# Dynamic Time Warping

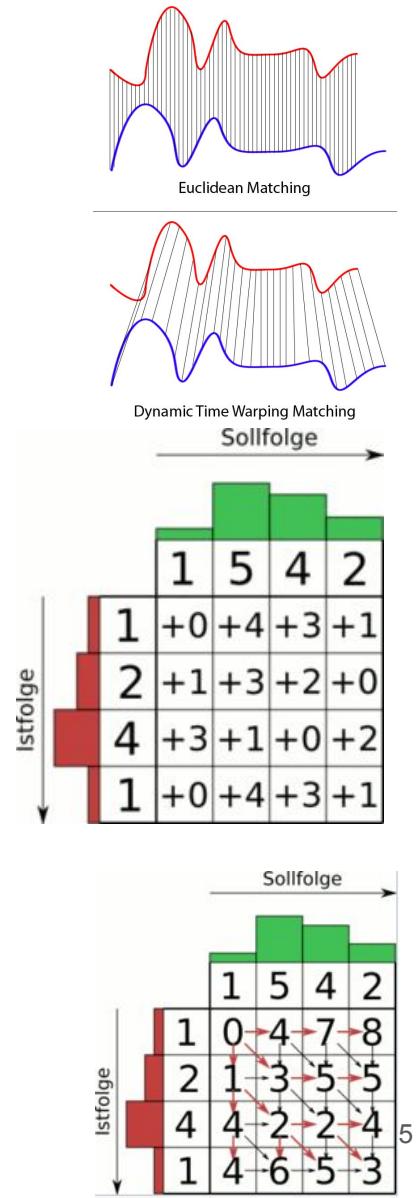
- Take two sequences of lengths N and M
- Build  $N \times M$  matrix  $\mathbf{d}$  of distances (diffs)
- Build a matrix of deformation,

$$D_{i,j} = d_{i,j} + \min(D_{i-1,j}, D_{i-1,j-1}, D_{i,j-1}). \quad (3)$$

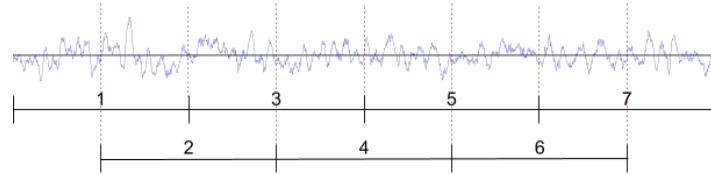
- Search for a path  $(1,1) - (N,M)$  with minimal average value weight.

$$DTW(Q, C) = \min \left\{ \frac{\sum_{k=1}^K d(w_k)}{K} \right\}. \quad (4)$$

Can give false positives



# Google Hum to search

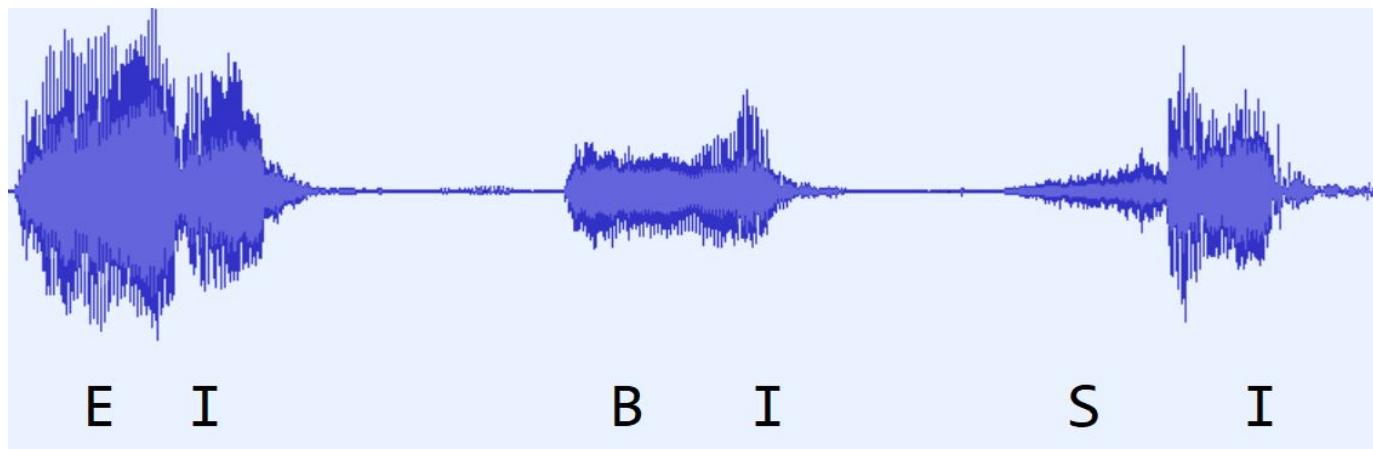


- Convolutional net to build a fingerprint-based from 8 seconds with 0.5 sec. step
- Inaccurate vector search with space partitioning and vector quantization
- Retrieve all candidate's fingerprints
  - Accurate match on the whole set of candidates' embeddings

# Speech (to text) processing

# Acoustic model

As text consist of letters, speech consists of phonemes.



AM: spectrum → phoneme

## Language model (in recognition)

Probabilistic model that predicts probability of a word given a sequence of phonemes.

Similar model is used to model sentences of words.

# Speech generation

- 1) Text preprocessing
  - a) Number to text
  - b) Abbreviations to text
  - c) Typo fix
- 2) Split text into phrases (punctuation, constructions)
- 3) Phonetic construction (language model)
  - a) queue - [kju]
  - b) Арбалетчиков
    - i) a0 r b a0 lj e1 t ch i0 k o0 v

# Speech generation

- 1) **Accents** are set
  - a) Using a dictionary
  - b) Using rules
  - c) Using statistics (speaker examples)
- 2) **Reversed acoustic model** is used to consider surrounding
- 3) **Timbre** is generation with **vocoder**
  - a) or RNNs



# Images search

Stanislav Protasov

# Agenda

- How our eyes work
- Historical approach to images search
- Duplicates search and CBIR
- Image and video understanding

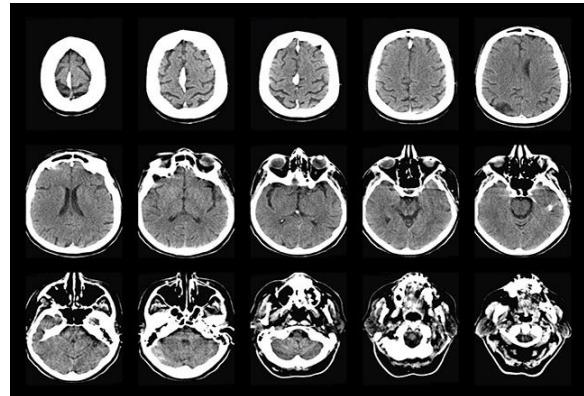
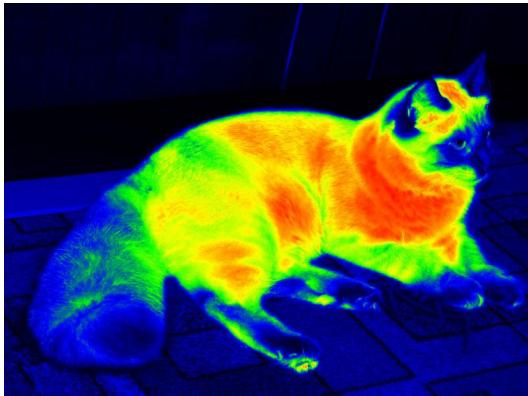
# How our vision works

***Hint:*** very similar to digital camera

# Vision

Vision is a sensor system, that receives information using **electromagnetic waves** [of visible spectrum].

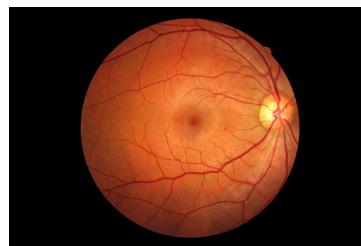
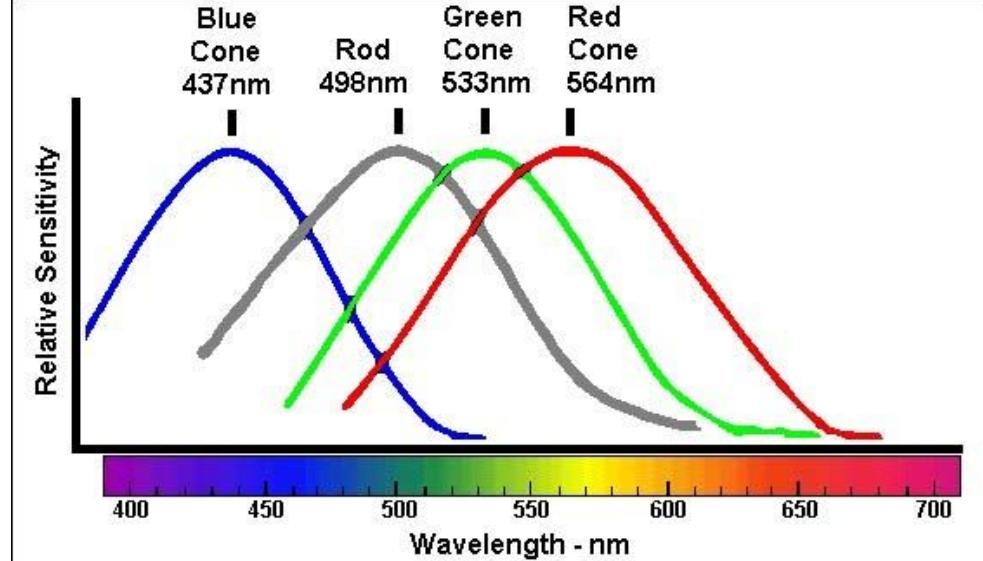
In general, X-ray, infrared and CT can be considered as “vision”.

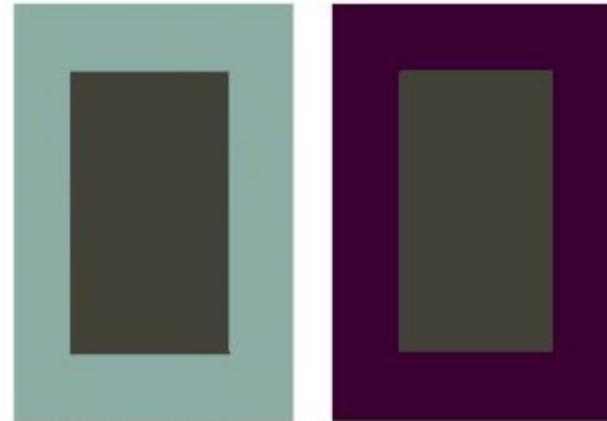
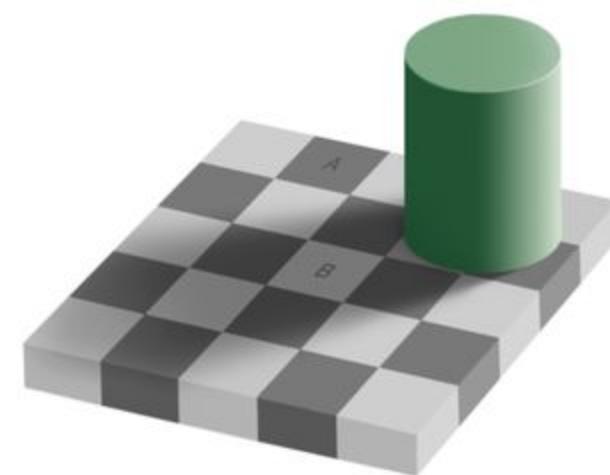


# Human vision

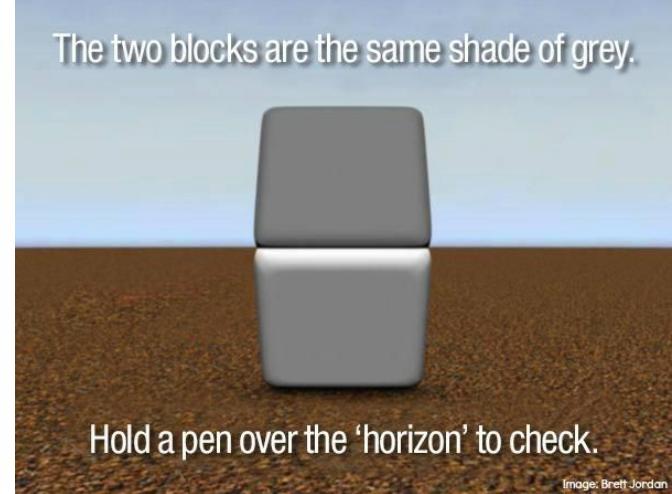
Major facts about vision:

- **Binocular** — allows restore 3D
- **Retina** — discrete
- **Color** — quantized
  - 4 types of sensor cells:
    - S,M,L-cone cells
    - Rod cells
- **Polarization and phase insensitive**
- Supports **focus**
- **Opponent-process theory** and
- **Color constancy**
  - Brain process differences of colors

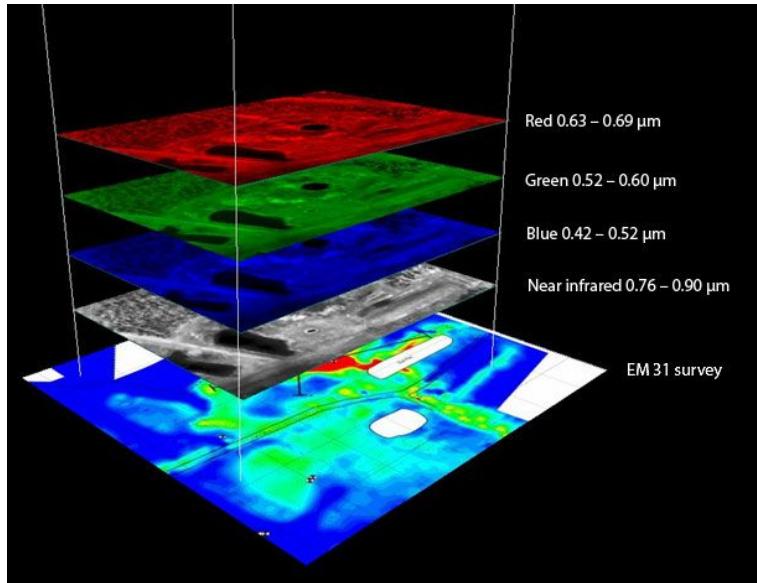
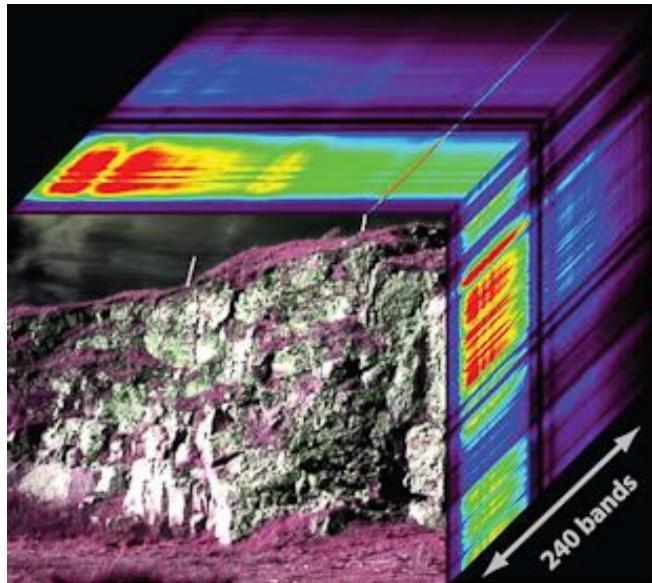




The two blocks are the same shade of grey.



# Multi- and hyperspectral images



# What is digital image

Digital image is a *quantized* and *discrete* vector field (similar to human vision). Each vector component describes:

- How much **energy is reflected** in particular spectrum part
  - Images, infrared images, ...

*OR*

- How much **energy is absorbed**
  - Medical imaging (X-ray, CT)

# How images are (were) retrieved

# Neighbouring text and subtitles



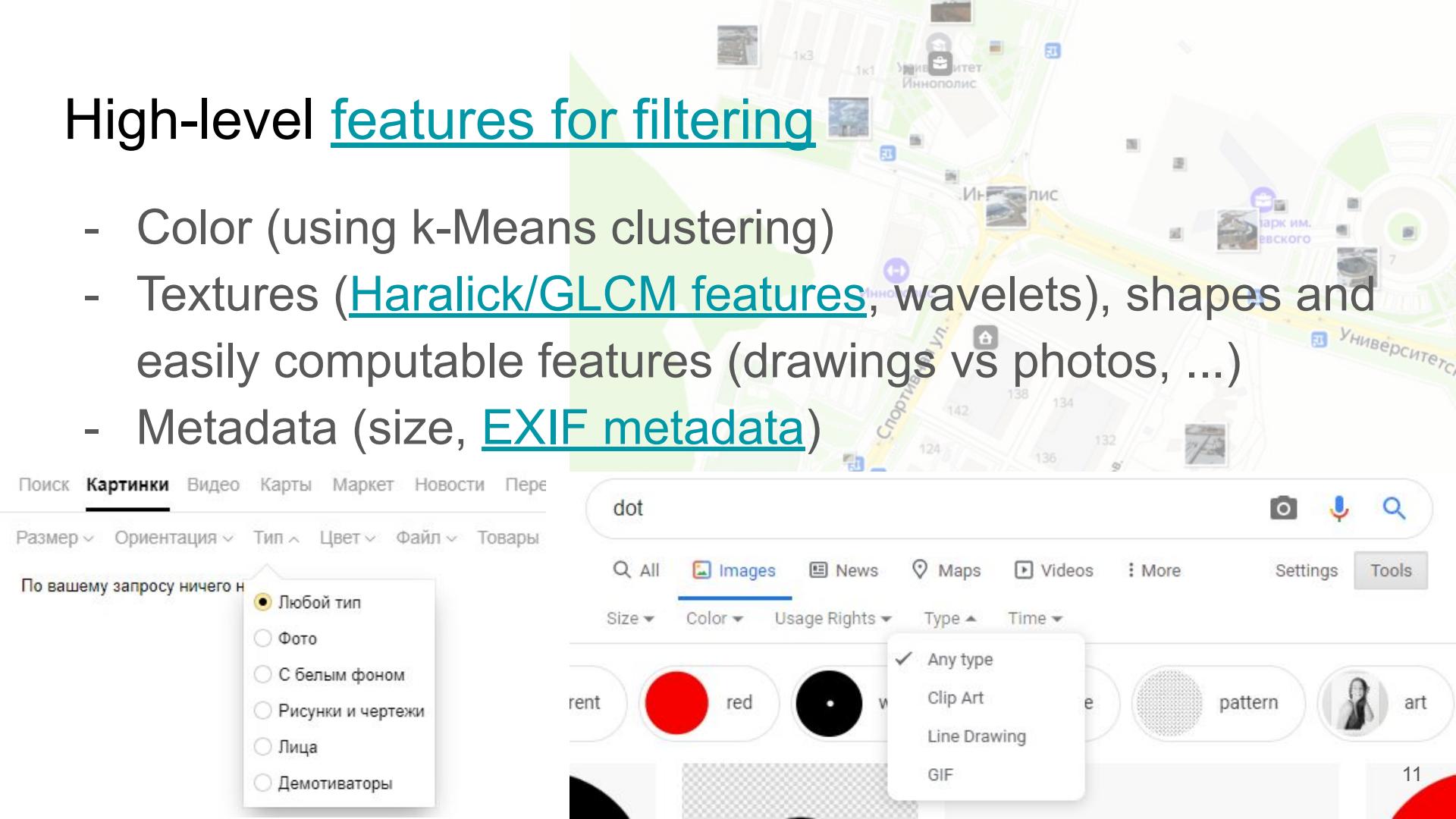
The muscles of the head allow the  
hoopoe's bill to be opened when it is  
inserted into the ground

the male.<sup>[4]</sup>

```
<div class="thumbinner" style="width:222px;">
  <a href="/wiki/File:Common_Hoopoe_(Upupa_epops)_at_Hodal_I_IMG_9225.jpg" class="image">
    <img alt="//upload.wikimedia.org/wikipedia/commons/thumb/2/25/Common_Hoopoe_%..MG_9225.jpg/220px-Common_Hoopoe_%28Upapa_epops%29_at_Hodal_I_IMG_9225.jpg" decoding="async" width="220" height="140" class="thumbimage" srcset="//upload.wikimedia.org/wikipedia/commons/thumb/2/25/Common_Hoopoe_%..MG_9225.jpg/330px-Common_Hoopoe_%28Upapa_epops%29_at_Hodal_I_IMG_9225.jpg 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/2/25/Common_Hoopoe_%..MG_9225.jpg/440px-Common_Hoopoe_%28Upapa_epops%29_at_Hodal_I_IMG_9225.jpg 2x" data-file-width="800" data-file-height="508"> == $0
  </a>
  <div class="thumbcaption">
    ><div class="magnify">...</div>
    "The muscles of the head allow the hoopoe's bill to be opened when it is inserted into the ground"
  </div>
</div>
```

# High-level features for filtering

- Color (using k-Means clustering)
- Textures (Haralick/GLCM features, wavelets), shapes and easily computable features (drawings vs photos, ...)
- Metadata (size, EXIF metadata)



**CBIR** = Content Based Image Retrieval

# CBIR

## Problems (sensitivity increases)

- Similarity search
- Duplicate search
- Identification (exactly the same, but with respect to e.g. compression)

# Similarity and duplicate search: image as a *bag of words*

In CV ... a **feature [point]** is defined as  
an "interesting" part of an image.

Usually for **interesting points** consider:

- Edges
- Corners
- Regions

After detector *feature vector (descriptor)*  
is computed.

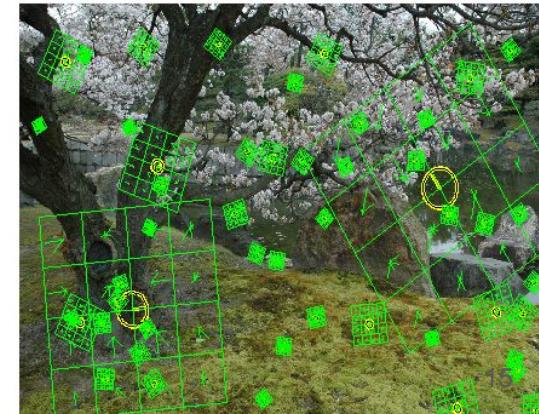
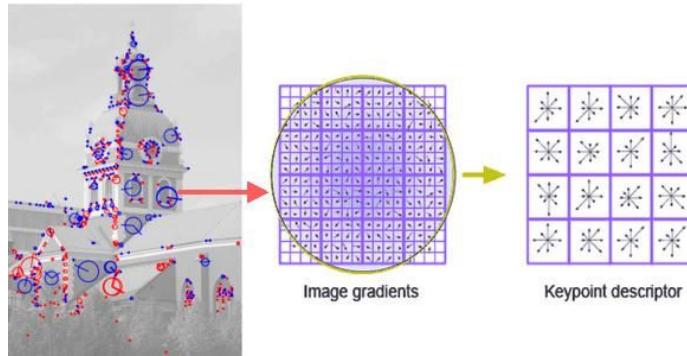


Use feature vector sets to describe **objects**

$$\triangle[G_\sigma(x,y) * f(x,y)] = [\triangle G_\sigma(x,y)] * f(x,y) = LoG * f(x,y)$$

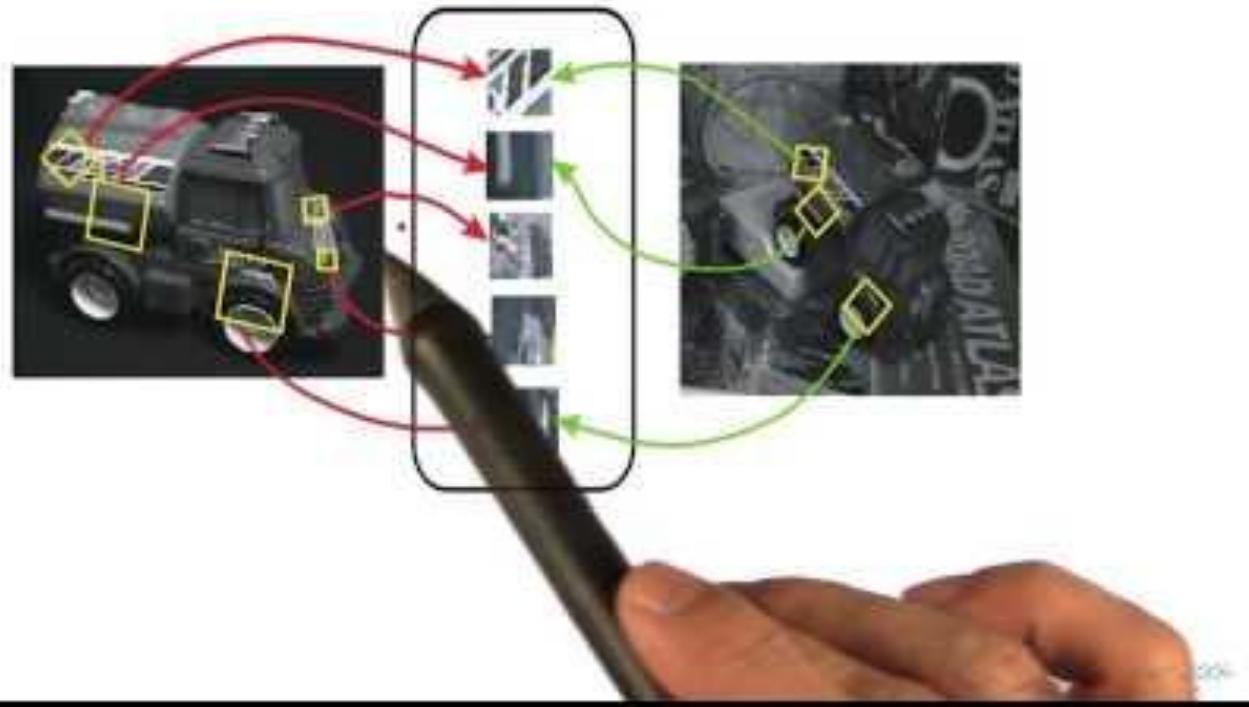
# SIFT: Scale-invariant feature transform

- 1) Compute gradients for images in *image pyramid* using difference of Gaussians (DoG). (*Image pyramid ~ Scale invariant*)
- 2) Search for local extrema in scale and space (*keypoints*)
- 3) Compute *direction* (*rotation invariant*)
- 4) Create descriptor: in 16x16 neighbourhood make 16 blocks, compute gradients (8 bins for angles) and make a vector.
- 5) Normalize (*intensity invariant*)



# SIFT overview

## Invariant Local Features



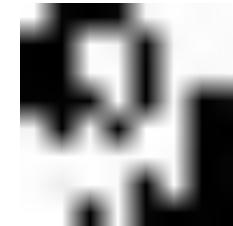
# Image fingerprinting for **duplicate search**

1. Use PoI. Allows cropping, need ~100 points, fails for texts
2. Use hash functions:
  - a. Image.Match based on Xerox features
    - Grayscale color image
    - Place 9x9 uniform grid of pixels
    - Each point is described with 8-neighbourhood {darker = -2, mild darker , ... , lighter = +2 }
    - Concatenate

# Image fingerprinting for duplicate search (2)

- Hash functions ([pip install ImageHash](#)):
- [\[average\] aHash](#)

- Resize to 8x8
- Grayscale
- Binarize by average
- Use Hamming dist



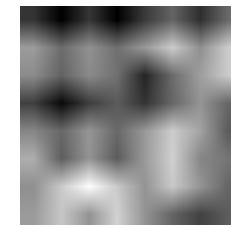
- [\[perception\] pHash](#) and [\[wavelet\] wHash](#)

- pHash uses DCT
- wHash - DWT, both coarse grained
- Use Hamming dist



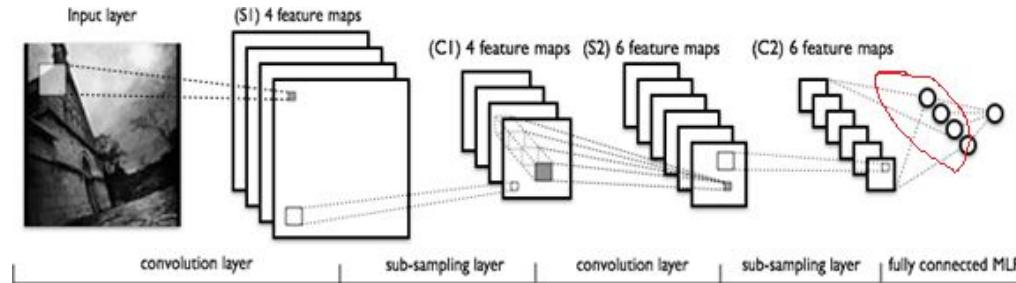
- [\[difference\] dHash](#)

- Resize to 9x8
- Grayscale
- Compute  $I[x+1, y] \leftrightarrow I[x, y]$  and use this as a bit

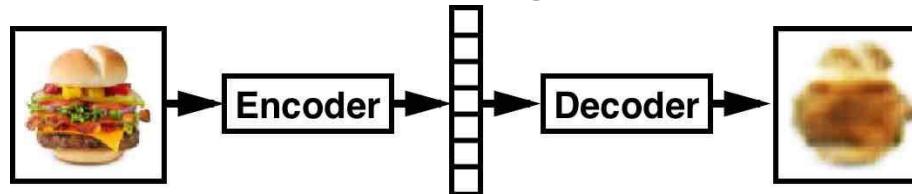


# Deep networks for specific and general **similarity** search

1. Images are of **different types** (classes, e.g. ImageNet). [Train classification network](#) (AlexNet, VGG16, ...) and use embeddings (from inner layer) as index.



2. Images are of the same type (faces). Train deep [convolutional autoencoder](#) which creates small-dimensional embeddings.



# Image understanding, video structure

# Semantic retrieval

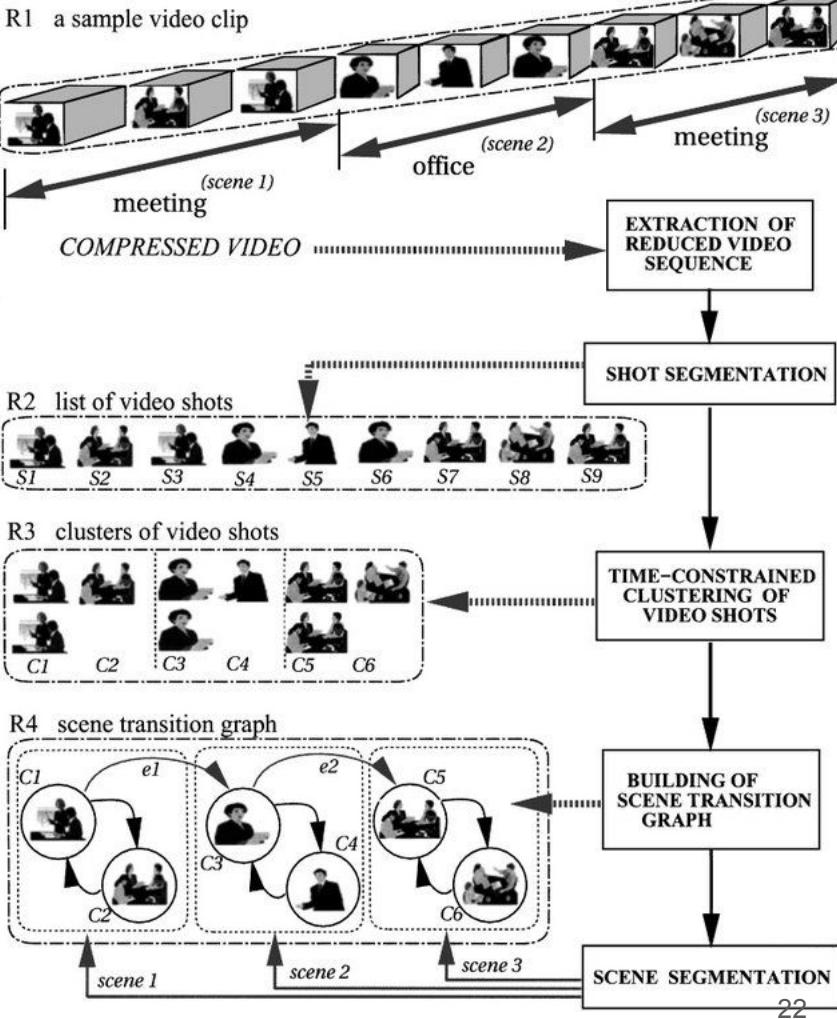
Deep classification and region-based networks allow adding semantic indices.

*NB:*

- *How many \$\$ will single inference cost for 20B of images?*
- *How much time?*
- *How often should a company do it?*

# Video structure mining

As text can be searched for a **paragraph**,  
Long videos should be also indexed with  
**scenes**. [[demo](#)]



# Probability power: language and topic modelling

Stanislav Protasov

# Agenda

- First, mostly of **language modelling**
- And then, little on **topic modelling**

# Language Model: beginning

**Language model** is used to *generate* texts (e.g. suggest next word) or to compare texts being similar in some case (“*recognize*”)

# Prerequisites

What is mathematical **model**?

Why do we need **model**? (in general)

Which **models** did we already consider, and how did we **benefit**:

- Vector Space Model
- Deep ANNs
- Small World Model
- Regular languages for syntax modelling
- ...

# What is a language

Language is a set. **Language model** serves for two tasks:

- **Guarding predicate**  $L = \{w \mid P(w)\}$  # recognition
- **Set generation procedure**

And as any other model, it helps to solve tasks stated for its prototype (real language):

- Are these words similar?
- What is the answer for the question?
- Are those texts from the same topics?
- ...
- **propose your own tasks**

# Did we already cover any **language models**?

- Vector space and latent space model (\*)
- Chomsky hierarchy models...
  - including finite state automata

# Very simple model (from the book)

This is a sum ...

$$\sum_{s \in \Sigma^*} P(s) = 1$$

This is a language set (all sentences)

$$P_{\text{uni}}(t_1 t_2 t_3) = P(t_1)P(t_2)P(t_3).$$

# Unigram model

UM — token (word) frequency as probability estimation

$$P(s) =$$

$$= P(\text{"abbcccd"}) = P(\text{"abbcccd[STOP]"}) =$$

$$= P(\text{"[\sim S]a[\sim S]b[\sim S]b[\sim S]c[\sim S]c[\sim S]c[\sim S]d[STOP]"}) =$$

$$= [ P(\text{"a"}) * P(\text{"b"})^2 * P(\text{"c"})^3 * P(\text{"d"}) ] * \\ [ P(\text{~STOP})^7 * P(\text{STOP}) ]$$

Constant for sentences of equal length

$$P(\text{query}) = \prod_{\text{term in query}} P(\text{term})$$

# Unigram model

Consider  $\{a, b\}$  as vocabulary.

$$P(\$) = \frac{1}{2}$$

$$P(a\$) = P(b\$) = \frac{1}{2} * \frac{1}{2}^2 = \frac{1}{8} \quad | \frac{1}{4}$$

$$P(aa\$) = P(ba\$) = \frac{1}{2} * \frac{1}{2} * \frac{1}{2}^3 = \frac{1}{32} \quad | \frac{1}{8}$$

In practice **constant part** is omitted:

1. For same length this is constant
2. In search we compare models. For model comparison constant factor is omitted

# Model types

General model

$$P(t_1 t_2 t_3 t_4) = \underline{P(t_1)} P(t_2 | t_1) P(t_3 | t_1 t_2) P(t_4 | t_1 t_2 t_3)$$

Unigram model

$$P_{\text{uni}}(t_1 t_2 t_3 t_4) = \underline{P(t_1)} P(t_2) P(t_3) P(t_4)$$

Bigram model

$$P_{\text{bi}}(t_1 t_2 t_3 t_4) = \underline{P(t_1)} P(t_2 | t_1) P(t_3 | t_2) P(t_4 | t_3)$$

How to find  $P(t_1)$ ,  $P(t_2 | t_1)$ ?

# Unigram model under the microscope == multinomial distribution

We don't care about the order, that is why sentence is a bag of words:

Professor called Stas ate a dog == Dog called Stas ate a professor

Probability of bag  $d$  then is described by a multinomial distribution:

$$P(d) = \frac{L_d!}{\text{tf}_{t_1,d}! \text{tf}_{t_2,d}! \cdots \text{tf}_{t_M,d}!} P(t_1)^{\text{tf}_{t_1,d}} P(t_2)^{\text{tf}_{t_2,d}} \cdots P(t_M)^{\text{tf}_{t_M,d}}$$

Here,  $L_d = \sum_{1 \leq i \leq M} \text{tf}_{t_i,d}$  is the length of document  $d$ ,  $M$  is the size of the term vocabulary, and the products are now over the terms in the vocabulary, not the positions in the document.

# Language Model: how to use it in IR

Language model is another way of **comparing texts**. For this you can use language models of query and text. And look for such text, which are “better” in generating queries.

Use **KL-divergence** for this.

# Ranking, oh yeah!

Likelihood

$$L(\theta \mid x) = p_\theta(x) = P_\theta(X = x)$$

Query Likelihood Model     $L(q_{\text{parameter}} \mid d_{\text{var}})$

$$P(d|q) = P(q|d)P(d)/P(q)$$

$$P(d \mid q) \sim P(q \mid d)$$

$$P(d) = \frac{L_d!}{\text{tf}_{t_1,d}! \text{tf}_{t_2,d}! \cdots \text{tf}_{t_M,d}!} P(t_1)^{\text{tf}_{t_1,d}} P(t_2)^{\text{tf}_{t_2,d}} \cdots P(t_M)^{\text{tf}_{t_M,d}}$$

Each text is a language! ლ(ಠ益ಠლ)

1. Build a model for each text (does it look similar to building TDM?)
2. Compute relevance for each doc (how **likely** this **query** is generated **by the doc language**)

$$P(q|M_d) = K_q \prod_{t \in V} P(t|M_d)^{\text{tf}_{t,d}}$$

But wait, we have ***the*** language model  $M_c$

1. If  $\hat{P}(t|M_d) = 0$  ( $\text{tf}_{t,d} = 0$ )  $\hat{P}(t|M_d) \leq \text{cf}_t/T$

2. Linear interpolation

$$\hat{P}(t|d) = \lambda \hat{P}_{\text{mle}}(t|M_d) + (1 - \lambda) \hat{P}_{\text{mle}}(t|M_c)$$

3. Bayesian smoothing  $\hat{P}(t|d) = \frac{\text{tf}_{t,d} + \alpha \hat{P}(t|M_c)}{L_d + \alpha}$

# Summary

1. Unigram model  $P_{\text{uni}}(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$
2. Unigram max likelihood query  $L(d|q) \sim P(q|d)$
3.  $P(t|d) \neq \text{tf}_{t,d} / D$  as it leads to 0-s
4. Estimate using **global language model**

$$P(d|q) \propto P(d) \prod_{t \in q} ((1 - \lambda)P(t|M_c) + \lambda P(t|M_d))$$

# Extended LM and motivations

Add **query language model** and use Kullback-Leibler divergence

$$R(d; q) = KL(M_d \| M_q) = \sum_{t \in V} P(t|M_q) \log \frac{P(t|M_q)}{P(t|M_d)}$$

What about **multilingual retrieval**? Just add translation matrix!

$$P(q|M_d) = \prod_{t \in q} \sum_{v \in V} P(v|M_d) T(t|v)$$

# Topic modelling

There are (latent) topics, which define behaviour of language models. **Text belongs to a combination of few topics**, which can be discovered using TM.

# Topic model

## Purpose:

1. Which **topics** a **document** belongs to
2. Which **words** form which **topic**

## Which questions answer:

1. How did the topic evolve in media?
2. What are major topics of this author?
3. How many topics are there?
4. ...

# Conditional independence hypothesis

How **words are generated** in the model does not depend on ~~particular document~~, but **on the topic** of the document.

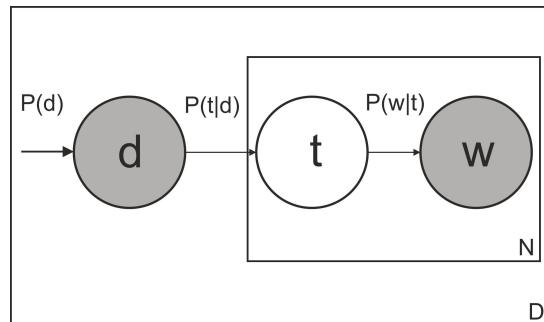
$$p(w | d, t) = p(w | t);$$

$$p(d | w, t) = p(d | t);$$

$$p(d, w | t) = p(d | t)p(w | t).$$

# Topic model is (formally)

$$p(d, w) = \sum_{t \in T} p(t)p(w|t)p(d|t) = \sum_{t \in T} p(d)p(w|t)p(t|d) = \sum_{t \in T} p(w)p(t|w)p(d|t)$$



$$p(w | d) = \sum_{t \in T} p(t | d) p(w | t).$$

**$p(w|d)$  - how to estimate and model these values?**

$$\Phi = ||p(w|t)||$$

$$\Theta = ||p(t|d)||$$

# Looking for decomposition $F \approx \Theta\Phi!$

**Sparseness hypothesis:** document belongs to a very limited topic number. Thus, both  $\Theta$  and  $\Phi$  are sparse.

They model **probabilities**, thus,  $p(*)|* \in [0, 1]$  and

$$\sum_w p(w|t) = 1, \quad \sum_t p(t|d) = 1, \quad \sum_t p(t) = 1,$$

**SVD and PCA don't work** (non-stochastic matrix + bias)

Scientists argue whether **PLSA** ([Vorontsov](#)) or **LDA** ([Blei, Ng](#)) is better

$B(y|e)!$