Technical Presentation

Pavel Aitkulov

ajtkulov@gmail.com

March 22, 2024

About me

- software engineer
- scala, fp, reasonable fp
- chess, Elo = 2000..2100, max = 2118
- run, marathon: 3h 22m, $\frac{1}{2}$ marathon: 1h 38m, 10km: 40m 40s, 5km: 19m 48s
- billiard
- ChGK, What? Where? When?, team quiz

Agenda

Best project?

- most challenging
- biggest business impact
- rps, highload, etc
- social impact

Agenda

Best project?

- most challenging
- biggest business impact
- rps, highload, etc
- social impact
- **information gain**

Agenda

Best project?

- most challenging
- biggest business impact
- rps, highload, etc
- social impact
- **information gain**

It's about embeddings.

Spell checker

Domain: company info, firmographics, B2B.

55-60M companies.

We have a search API. Enhance your data in CRM.

There are a lot of misspellings/typos in company names (human errors, legacy systems).

Spell checker, challenge

Different languages, artificial names (Uber, Google, everyone wants to be unique).

Spell checker, challenge

Different languages, artificial names (Uber, Google, everyone wants to be unique).

- 9M unique words
- 50M unique pairs
- 65M unique triples

We don't want to drop long tail.

Spell checker, challenge

Different languages, artificial names (Uber, Google, everyone wants to be unique).

- 9M unique words
- 50M unique pairs
- 65M unique triples

We don't want to drop long tail.

Frequencies: *Apple INC* $\rightarrow$ *Aapple INC*, which one is more natural?

Spell checker, similarity

Similar to auto-suggestion in Search Engine, in Word/Google Docs.

Specific domain.

There are OSS solutions for 300K words, without frequencies.

Solr/Lucene/Elastic has its own solutions. Limited.

Spell checker, similarity

Similar to auto-suggestion in Search Engine, in Word/Google Docs.

Specific domain.

There are OSS solutions for 300K words, without frequencies.

Solr/Lucene/Elastic has its own solutions. Limited.

Natural language:

- 2-5K words daily
- 50-200K vocabulary

Spell checker, input

Company names, 55M.

API logs + output size, history, 6 months.

Every 15-17% request contains a new word, that doesn't exist in the company's name set.

20-50% of those queries could be fixed.

Depending on the client, we had a 20-30% hit rate (precision).

Spell checker, typos types

- Classic misspelling, insert/replace/delete.

  Carme Fresca Restaurante $\rightarrow$ *Carne Fresca Restaurante*

Spell checker, typos types

- Classic misspelling, insert/replace/delete.

  Carme Fresca Restaurante $\rightarrow$ *Carne Fresca Restaurante*

- Cut, last word:
  Carne Fresca Resta $\rightarrow$ *Carne Fresca Restaurante*

Spell checker, typos types

- Classic misspelling, insert/replace/delete.

  Carme Fresca Restaurante $\rightarrow$ *Carne Fresca Restaurante*

- Cut, last word:
  Carne Fresca Resta $\rightarrow$ *Carne Fresca Restaurante*

- No spaces: *carnefrescarestaurante* $\rightarrow$ *Carne Fresca Restaurante*

Spell checker, typos types

- Classic misspelling, insert/replace/delete.

  Carme Fresca Restaurante $\rightarrow$ *Carne Fresca Restaurante*

- Cut, last word:
  Carne Fresca Resta $\rightarrow$ *Carne Fresca Restaurante*

- No spaces: *carnefrescarestaurante* $\rightarrow$ *Carne Fresca Restaurante*

It is better to have context, at least 2 words.

Spell checker, typos types

- Classic misspelling, insert/replace/delete.

  Carme Fresca Restaurante $\rightarrow$ *Carne Fresca Restaurante*

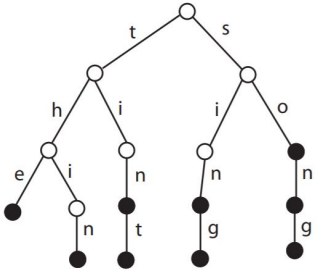
- Cut, last word:
  Carne Fresca Resta $\rightarrow$ *Carne Fresca Restaurante*

- No spaces: *carnefrescarestaurante* $\rightarrow$ *Carne Fresca Restaurante*

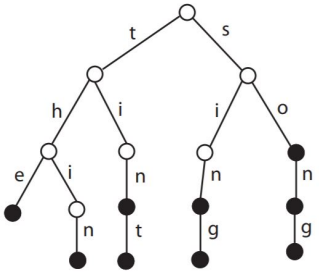
It is better to have context, at least 2 words.

Identification of error, based on freq. If $freq(word) \leq 5 \,||\, freq(word_1\ word_2) \leq f_2$.

Spell checker, Trie

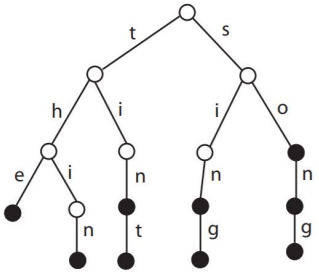


Spell checker, Trie



Classic algorithm to fix Levenshtein-distance errors. Looking for *sun*, *son*, *sin* has 1 replacement, *sing*, *song* has 1 replacement and 1 insert.

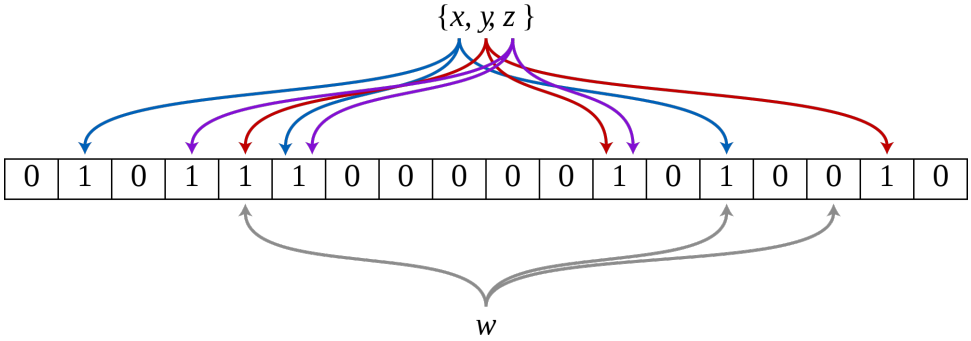
Spell checker, Trie



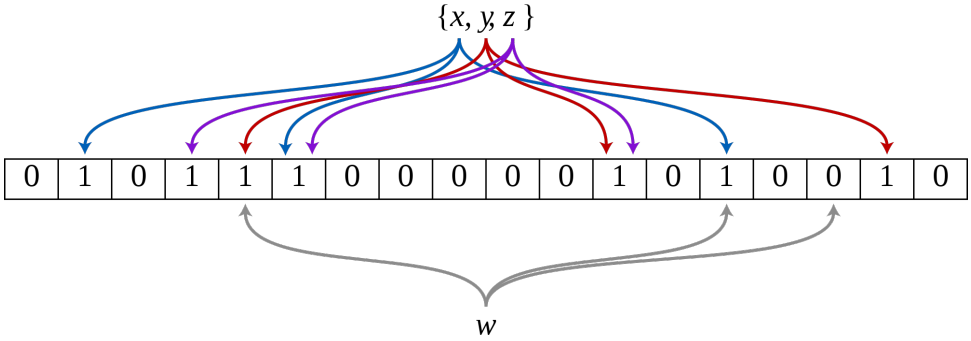
Classic algorithm to fix Levenshtein-distance errors. Looking for *sun*, *son*, *sin* has 1 replacement, *sing*, *song* has 1 replacement and 1 insert.

Compressed version. How do you store 9M unique and 50M pairs?

Spell checker, Bloom Filter



Spell checker, Bloom Filter



Compact HashTable (contains?), controlled FalsePositve-error, $10^{-6} \ldots 10^{-12}$, 10-30 bits per entity (1-3 bytes).

Spell checker, Trie as Bloom Filter

Trie as interface:

- contains a prefix
- contains a whole word, termination
- links to next level

Spell checker, Trie as Bloom Filter

Trie as interface:

- contains a prefix
- contains a whole word, termination
- links to next level

Boolean queries = Two Bloom Filters.

Spell checker, Trie as Bloom Filter

Trie as interface:

- contains a prefix
- contains a whole word, termination
- links to next level

Boolean queries = Two Bloom Filters.

Links to the next level = we have only 26 letters in English alphabet.

Optimization: use n-grams stats, there is no 'QZXP' in a dataset.

Spell checker, Data Structures, Algorithms

Trie as Bloom Filter. For 1-, 2-word ngrams. (actually, 3- was useless)

Frequencies as Count-Min-Sketch for 1-, 2-ngrams.

Spell checker, Data Structures, Algorithms

Trie as Bloom Filter. For 1-, 2-word ngrams. (actually, 3- was useless)

Frequencies as Count-Min-Sketch for 1-, 2-ngrams.

Levenshtein-distance errors.

Cut and no-space tasks (a good task for interview).

Spell checker, Data Structures, Algorithms

Trie as Bloom Filter. For 1-, 2-word ngrams. (actually, 3- was useless)

Frequencies as Count-Min-Sketch for 1-, 2-ngrams.

Levenshtein-distance errors.

Cut and no-space tasks (a good task for interview).

We are trying to fix only 1 error, otherwise there are too many results.

Extra math to evaluate results. Smoothing. Get top-5 results.

Spell checker, Data Structures, Algorithms

Trie as Bloom Filter. For 1-, 2-word ngrams. (actually, 3- was useless)

Frequencies as Count-Min-Sketch for 1-, 2-ngrams.

Levenshtein-distance errors.

Cut and no-space tasks (a good task for interview).

We are trying to fix only 1 error, otherwise there are too many results.

Extra math to evaluate results. Smoothing. Get top-5 results.

$$p(w_1 w_2 w_3 \dots w_n) = p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_1 w_2) \cdot \dots p(w_n|w_1 w_2 \dots w_{n-1})$$
$$\approx p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_2) \cdot \dots p(w_n|w_{n-1})$$

Spell checker, final implementation

Orb Intelligence, as a startup. 4-5 engineers, 1 ML person, 3 ML-interns, CEO, 1 sales, 1 team for ML markup.

Integration with Search API, python backend + Solr index (2 - 100 replicas).

Spell checker service, 1 instance, all indices in 20Gb RAM. Scala.

Spell checker, final implementation

Orb Intelligence, as a startup. 4-5 engineers, 1 ML person, 3 ML-interns, CEO, 1 sales, 1 team for ML markup.

Integration with Search API, python backend + Solr index (2 - 100 replicas).

Spell checker service, 1 instance, all indices in 20Gb RAM. Scala.

Bloom Filter in Guava/Algebird is bad. Fork of addthis/stream-lib. CMS is enough.
Hit rate 20-30%. With Spell checker we had an extra 3-5% hit rate. 10-100 rps, max 4K.

Spell checker, final implementation

Orb Intelligence, as a startup. 4-5 engineers, 1 ML person, 3 ML-interns, CEO, 1 sales, 1 team for ML markup.

Integration with Search API, python backend + Solr index (2 - 100 replicas).

Spell checker service, 1 instance, all indices in 20Gb RAM. Scala.

Bloom Filter in Guava/Algebird is bad. Fork of addthis/stream-lib. CMS is enough.
Hit rate 20-30%. With Spell checker we had an extra 3-5% hit rate. 10-100 rps, max 4K.

One SE for 3-4 months, I'm. Plus, one SE for 1-2 weeks for integration, logs + analytics. In 2020.

Spell checker, outcome

Idea

Business outcome

Spell checker, outcome

Idea

Business outcome

Acquisition by Dun and Bradstreet.

Search API was deprecated and shut down in 9 months later.

Spell checker, outcome

Idea

Business outcome

Acquisition by Dun and Bradstreet.

Search API was deprecated and shut down in 9 months later.

Trie as Bloom Filter. My personal idea, re-invented, there is an article from 2016 in bioinformatics with the same idea.

Duplicates detection

Dun and Bradstreet, B2B, 600M companies profiles.

We want to find similar profiles.

Company name + full address (address1, address2, city, state, country, with empty value).

Duplicates detection

Dun and Bradstreet, B2B, 600M companies profiles.

We want to find similar profiles.

Company name + full address (address1, address2, city, state, country, with empty value).

Wendy Coffee, 16 Park Square, London, England $\approx$ *Wendy's Coffee, 16N Park Square, London, England*

Duplicates detection

Dun and Bradstreet, B2B, 600M companies profiles.

We want to find similar profiles.

Company name + full address (address1, address2, city, state, country, with empty value).

Wendy Coffee, 16 Park Square, London, England $\approx$ *Wendy's Coffee, 16N Park Square, London, England*

Solution with $O(n^2)$ doesn't work.

Duplicates detection, Locality Sensitive Hashing

Classic hashing, equality of hash (almost) leads to equality on objects. Int32/Int64.

Duplicates detection, Locality Sensitive Hashing

Classic hashing, equality of hash (almost) leads to equality on objects. Int32/Int64.

Locality Sensitive Hashing (LSH). Family of methods.

$LSH(object) \rightarrow object_{simpler}$. Grid in multidimensional space, caps on a sphere, vector.

The similarity of LSH values leads to similarity in objects.

Duplicates detection, Locality Sensitive Hashing + MinHash

Convert text into a soup of n-grams; 2-3-4-length tokens, not words.

Convert soup of n-gram to binary vector via MinHash.

For each binary vector find all similar (by Hamming distance) vectors. See next slides

Duplicates detection, MinHash - 2

```
hashSet = ngrams.map(hash_function)

for idx in 0 until vector_size:
   permutation = genRandomPermutation(idx)
   output(idx) = min(permutation(hash_set))
```

Output: Array[Integer]

Number theory, permutation is defined by $(a, b) : f(x) = a \cdot x + b \, mod \, 2^{32}$. Only a is enough. We can apply $f^{-1}(x)$, doesn't matter.

Output: Array[Bit]. Apply $(x \rightarrow x \, mod \, 2)$

Reference: Mining of Massive Datasets Jure Leskovec, Anand Rajaraman, Jeff Ullman

Duplicates detection, ANN on binary vectors

Hamming Distance = Levenshtein Distance with replacement.

We solved it with Trie as Bloom Filter in the previous task.

The tricky part: binary vector + ID encoding.

Duplicates detection, KNN

Any solution with FAISS, HNSW, etc still works.

Pros/cons, no graphs, less memory, 20-30 bytes per item.

Duplicates detection, implementation details

600M company profiles, business registration name, brand name, multiple addresses. In total around 800M profiles.

Build index for all profiles. Traverse all other profiles against this index.

Index - 20Gb RAM.

24 nodes (32 GB RAM, 16 cpu), 1 week, Kafka for message bus, plain Scala.

Duplicates detection, results

Idea

Business outcome

Duplicates detection, results

Idea

Business outcome

6M similar profiles with different Hamming distance. To manually review, merge, or delete.

4M equal profiles, mostly, size 2 or 3. What went wrong: cluster 100K, should be preprocessed. $O(n^2)$ output. Duplicates could be automatically merged or deleted.

PoC state, one-off.

Dissernet

Dissernet is a volunteer project dedicated to the verification of research articles and Ph.D. theses, focusing on detecting plagiarism and data manipulation.

https://www.dissernet.org/

https://en.wikipedia.org/wiki/Dissernet

Dissernet, what we do

- Examination for text plagiarism
- scrutiny of data plagiarism
- detection of data manipulation
- identification of reference manipulation
- translation scrutiny from other languages

Dissernet, example

Таблица заимствований Что это такое?

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
141	**142**	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167													

Dissernet, example

Проверяемый текст

[стр. 134]

136 минающего устройства при естественной выборке команд непосредственно со счетчика 9 команд); 2) На втором такте к содержимому счетчика 9 команд прибавляется единица подготавливается адрес следующей команды; 3) На третьем такте сигналы микрокоманды поступают на вход считывания регистра 11 адреса и на вход считывания содержимого ячейки памяти запоминающего устройства по указанному адресу.
При этом команда, хранящаяся в первой ячейке памяти, записывается в регистр 12 числа; 4) На четвертом такте сигналы микрокоманды подаются на вход считывания регистра 12 числа, вход второго коммутатора 7 и на вход дешифратора 3 кода операции, где раскодируются, после чего управляющий узел 1 переходит ко второму этапу работы.
запи-Для примера рассмотрим порядок исполнения одной из команд, санной в регистре 12 числа после выполнения первых четырех тактов.
Пусть в поле кода операции команды содержимого регистра 12 числа записана команда сложения содержимого регистра 13 сумматора с числом расположенным запоминающем устройстве по адресу, указанному в поле адреса регистра 12 числа (при использовании одноадресной команды).
Управляющий узел 1 при этом выдает следующие микрокоманды: 5) на пятом такте сигналы микрокоманды подаются на вход считывания регистра 12 числа, на вход второго коммутатора 7, первого коммутатора 6 и на вход записи регистра 11 адреса (адрес, хранящийся в регистре 12 числа записывается в регистр 11 адреса, содержимое регистра 12 числа обнуляется); 6) На шестом такте сигналы микрокоманды подаются на вход считывания регистра 11 адреса, на вход первого коммутатора 6, на вход считывания запоминающего устройства и на вход записи регистра 12 числа (из запоминающего устройства в регистр 12 числа записывается второе слагаемое (считаем, что первое слагаемое уже находится в регистре 13 сумматора)).

[стр. 130]

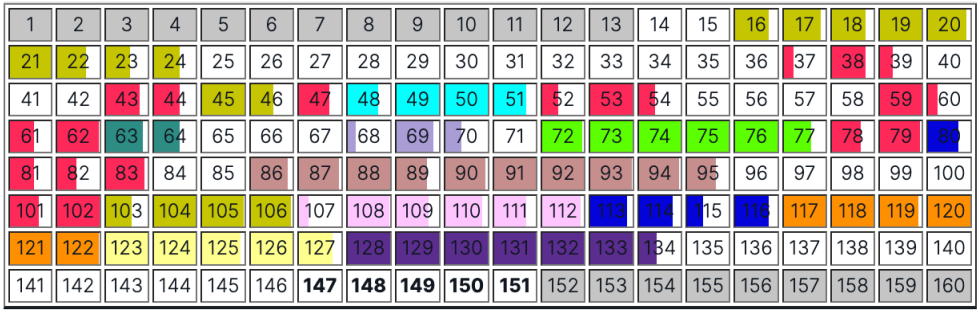
129 регистра 11 адреса, при этом содержимое счетчика 9 команд пересылается в регистр 11 адреса (или через первый коммутатор 6 на адресные входы запоминающего устройства при естественной выборке команд непосредственно со счетчика 9 команд); 2) На втором такте к содержимому счетчика 9 команд прибавляется единица подготавливается адрес следующей команды; 3) На третьем такте сигналы микрокоманды поступают на вход считывания регистра 11 адреса и на вход считывания содержимого ячейки памяти запоминающего устройства по указанному адресу.
При этом команда, хранящаяся в первой ячейке памяти, записывается в регистр 12 числа; 4) На четвертом такте сигналы микрокоман
подаются на вход считывания регистра 12 числа, вход второго коммутатора 7 и на вход дешифратора 3 кода операции, где
раскодируются, после чего управляющий узел 1 переходит ко второму этапу работы.
Для примера, рассмотрим порядок исполнения одной из команд, записанной в регистре 12 числа после выполнения первых четырех
тактов.
Пусть, в поле кода операции команды содержимого регистра 12 числа записана команда сложения содержимого регистра 13 сумматор
с числом, расположенным запоминающем устройстве по адресу, указанному в поле адреса регистра 12 числа (при использовании
одноадресной команды).
Управляющий узел 1 при этом выдает следующие микрокоманды: 5) на пятом такте сигналы микрокоманды подаются на вход
считывания регистра 12 числа, на вход второго коммутатора 7, первого коммутатора 6 и на вход записи регистра 11 адреса (адрес,
хранящийся в регистре 12 числа записывается в регистр 11 адреса, содержимое регистра 12 числа обнуляется); 6) На шестом такте
сигналы микрокоманды подаются на вход считывания регистра 11 адреса, на вход первого коммутатора 6, на вход считывания
запоминающего устройства и на вход записи регистра 12 числа (из запо

[стр.,131]

130 минающего устройства в регистр 12 числа записывается второе слагаемое (считаем, что первое слагаемое уже находится в регист
13 сумматора). 7) На седьмом такте сигналы микрокоманды подаются на вход считывания регистра 12 числа и регистра 13 сумматора
при этом арифметикологическое устройство осуществляет операцию сложения и запись результата сложения в регистр 13 сумматор
следующим образом.

Dissernet, example

-



Dissernet, example

Хадж Аисса Абделхак (2020)

Таблица 4 - Показатели центральной гемодинамики при клинических признаках преэклампсического синдрома у жеребых кобыл (n= 12)

Показатели	Группы	
	Клинически здоровые	Преэклamптический синдром
СГД, мм. рт. ст.	102,5±6,9	95,3±3,7*
ОПСС, дин. с. см^5	1572±104	1724±89**
ЧСС, уд/мин	86,4±3,2	116,7±3,1
Уо, мл	65,7±3,3	51,1±3,2*
МОК, мл/мин	6824±234	4022±98**
Си, л/мин, м^2	3,51±0,09	2,34±0,07*

63

Родин Павел Владимирович (2006)

Таблица 1

Показатели центральной гемодинамики у клинически здоровых сук и при фетоплацентарной недостаточности (n=23)

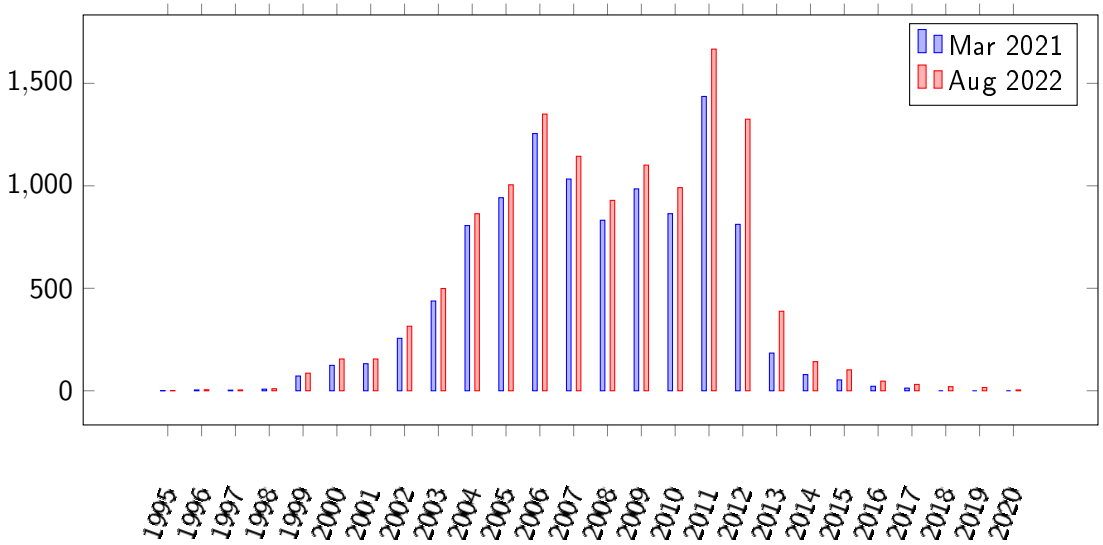
Показатели	Группы (M±m)	
	Клинически здоро-вые	Фетоплацентарная недостаточность
СГД, мм.рт.ст.	102,5±6,9	95,3±3,7*
ОПСС, дин. с. см^5	1572±104	1724±89**
ЧСС, уд/мин	86,4±3,2	86,7±3,1
Уо, мл	65,7±3,3	51,1±3,2*
МОК, мл/мин	6824±234	4022±98**
Си, л/мин, м^2	3,51±0,09	2,34±0,07*

Примечание: * - $p < 0,05$; ** - $p < 0,01$

Dissernet, results

- retraction of over 1400 Ph.D. diplomas
- inclusion of 12K Ph.D. works and over 50K research articles in our comprehensive database

Dissernet, PhD cases



Dissernet

Social impact

Tech challenge

Dissernet

Social impact

Tech challenge

Plain Scala. n-grams analysis, 500Gb of raw text, 2.6M articles, 600K dissertations.

Dissernet

Social impact

Tech challenge

Plain Scala. n-grams analysis, 500Gb of raw text, 2.6M articles, 600K dissertations.

Good task for interview.

QA