common search:

Ranking the Web with Spark

Apache Big Data Europe 2016

sylvain@sylvainzimmer.com
@sylvinus

/usr/bin/whoami

- Jamendo (Founder & CTO, 2004-2011)
- TEDxParis (Co-founder, 2009-2012)
- dotConferences (Founder, 2012-)
- Pricing Assistant (Co-founder & CTO, 2012-)

transparency

reproducibility

common search:

common search: python EN

About 41380 results

Welcome to **Python**.org
www.python.org
The official home of the **Python** Programming Language

Dive Into **Python**
www.diveintopython.net
This book lives at . If you're reading it somewhere else, you may not have the latest version.

The Eric **Python** IDE
eric-ide.python-projects.org
Eric is a full featured **Python** editor and IDE, written in **Python**. It is based on the cross platform Qt gui toolkit, integrating the highly flexible Scintilla...

Starship
www.python.net
The home of pythonistas

Tutorials, **Python** Courses: Online and On Site
python-course.eu
Free comprehensive online tutorials suitable for self-study and high-quality on site **Python** courses in Europe, Canada (Toronto) and the US

learning **python** | one man's journey into **python**…
www.learningpython.com
one man's journey into **python**...

https://uidemo.commonsearch.org

Common Search Explainer | Search test | URL test | About

python | OK | EN

Results (50)

Welcome to Python.org

[debug] https://www.python.org/

The official home of the Python Programming Language

docid -4478921722574158000

static rank 0.7923434

ES score 87.821815

ES explain

```
47.75143 | sum of:
    47.75143 | function score, product of:
        60.87483 | max plus 0.5 times others of:
            60.768433 | weight(domain_words:python in 77874) [PerFieldSimilarity], result of:
                60.768433 | score(doc=77874,freq=1.0 = termFreq=1.0
), product of:
                    8.0 | boost
                    5.97652 | idf(docFreq=9023, maxDocs=3555860)
                    1.2709827 | tfNorm, computed from:
                        1.0 | termFreq=1.0
                        1.0 | parameter k1
                        0.75 | parameter b
                        2.31778 | avgFieldLength
                        1.0 | fieldLength
            0.21279304 | weight(body:python in 77874) [PerFieldSimilarity], result of:
                0.21279304 | score(doc=77874,freq=21.0), product of:
                    0.14198984 | queryWeight, product of:
                        6.976686 | idf(docFreq=9021, maxDocs=3555860)
                        0.020352047 | queryNorm
                    1.4986497 | fieldWeight in 77874, product of:
                        4.582576 | tf(freq=21.0), with freq of:
                            21.0 | termFreq=21.0
                        6.976686 | idf(docFreq=9021, maxDocs=3555860)
                        0.046875 | fieldNorm(doc=77874)
        0.78441995 | min of:
            0.78441995 | function score, score mode [multiply]
                0.7923434 | function score, product of:
                    1.0 | match filter: *:*
```

https://explain.commonsearch.org/?q=python&g=en

Ranking

Disclaimer: IANASRE

(I Am Not A Search Relevance Engineer)

What's in a score

score = fn(doc, query, language, user, time)

What's in a score

score = fn(doc, query)

What's in a score

score = fn(static_score, dynamic_score (query))

Static score

Static features

- Scopes:
  - **Page:** URL depth, markup stats, ...
  - **Domain:** Age, page count, blacklists, ...
  - **WebGraph:** PageRank, ...

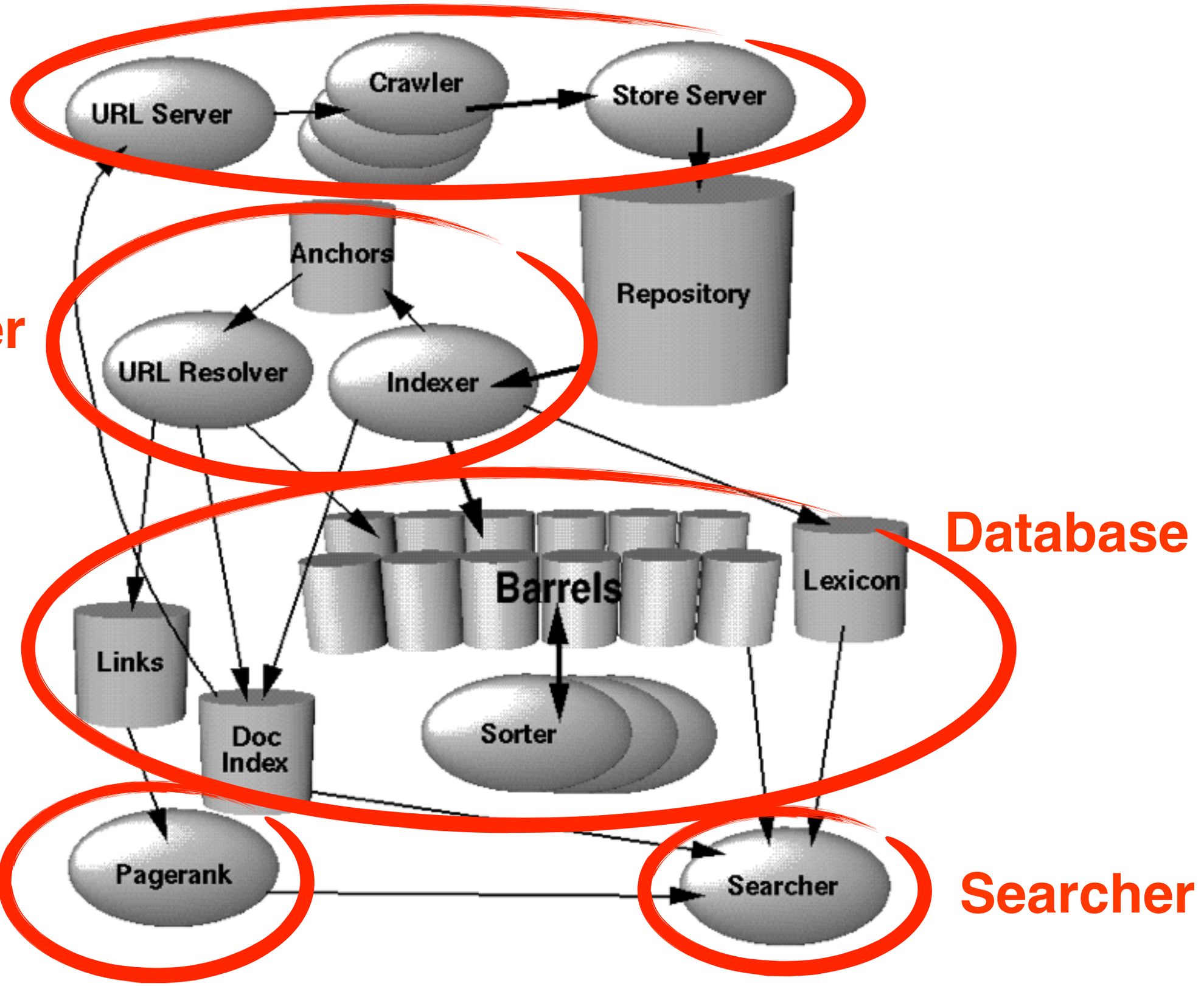
Crawler

Indexer

Database

Ranker

Searcher



The Anatomy of a Large-Scale Hypertextual Web Search Engine (1998)

http://infolab.stanford.edu/~backrub/google.html

Dynamic score

Dynamic features

- **Text match:** TF-IDF, BM25, proximity, topic, ...
- **Query-level:** number of words, popularity, ...
- **Usage:** clicks, dwell time, reformulations, ...
- **Time**

Scoring function

Data sources
Common Crawl, Alexa top 1M, ...

Indexer
Python, Spark

words, static score

Offline

Database
Elasticsearch

query

top 10 docs, final scores

Searcher
Go

Users

Online

```
{
    "query": {
        "function_score": {
            "query": {
                "bool": {
                    "must": {
                        "multi_match": {
                            "query": query_string,
                            "minimum_should_match": "-25%",
                            "type": "cross_fields",
                            "tie_breaker": 0.5,
                            "fields": ["title^3", "body", "url_words^2"]
                        }
                    }
                }
            },
            "functions": [{
                "field_value_factor": {
                    "field": "static_score",
                    "factor": 1
                }
            }]
        }
    }
}
```

Common Search Explainer | Search test | URL test | About

python | OK | EN

Results (50)

Welcome to Python.org

[debug] https://www.python.org/

The official home of the Python Programming Language

docid -4478921722574158000

static rank 0.7923434

ES score 87.821815

ES explain

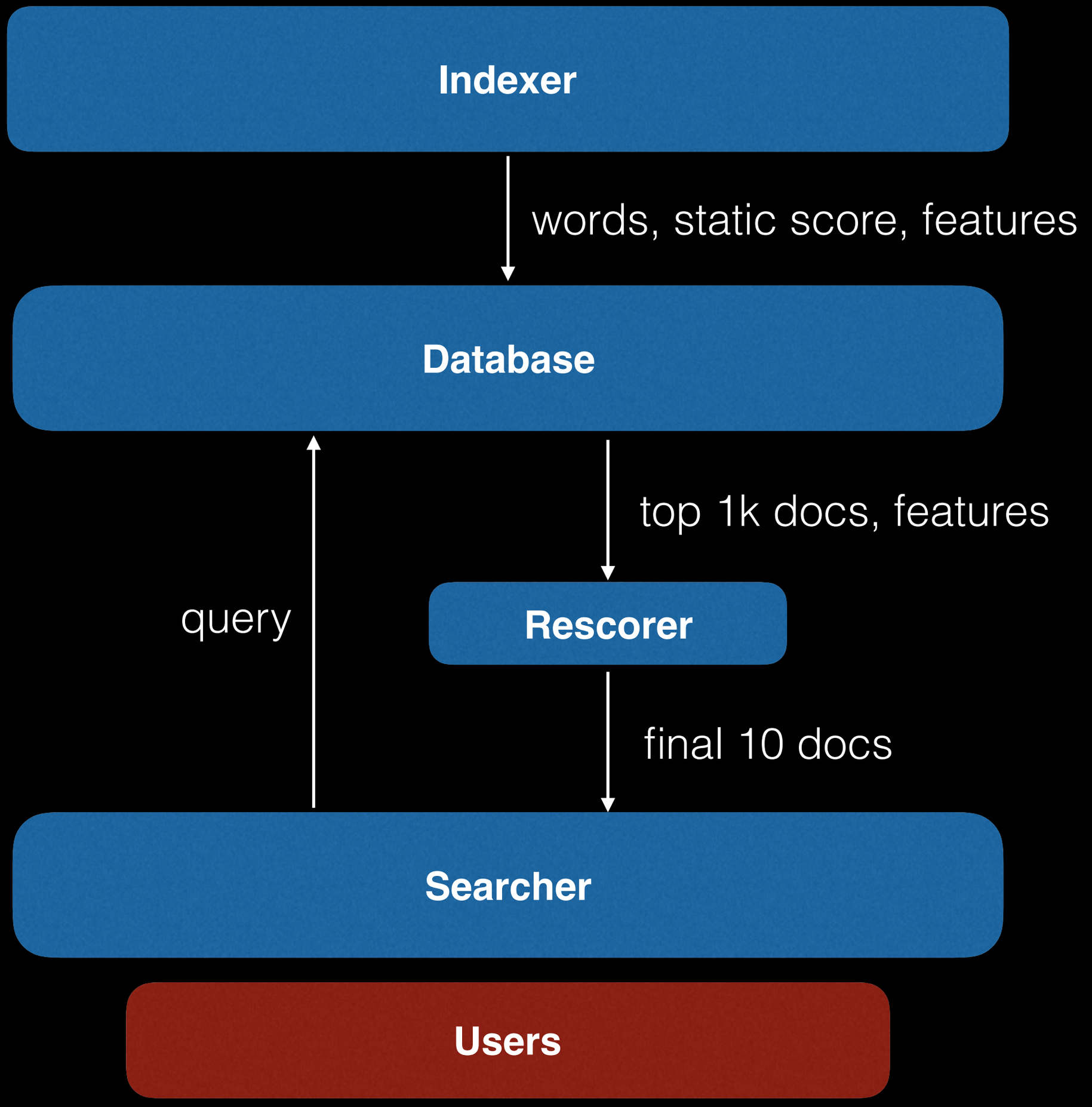
```
47.75143 | sum of:
    47.75143 | function score, product of:
        60.87483 | max plus 0.5 times others of:
            60.768433 | weight(domain_words:python in 77874) [PerFieldSimilarity], result of:
                60.768433 | score(doc=77874,freq=1.0 = termFreq=1.0
), product of:
                    8.0 | boost
                    5.97652 | idf(docFreq=9023, maxDocs=3555860)
                    1.2709827 | tfNorm, computed from:
                        1.0 | termFreq=1.0
                        1.0 | parameter k1
                        0.75 | parameter b
                        2.31778 | avgFieldLength
                        1.0 | fieldLength
            0.21279304 | weight(body:python in 77874) [PerFieldSimilarity], result of:
                0.21279304 | score(doc=77874,freq=21.0), product of:
                    0.14198984 | queryWeight, product of:
                        6.976686 | idf(docFreq=9021, maxDocs=3555860)
                        0.020352047 | queryNorm
                    1.4986497 | fieldWeight in 77874, product of:
                        4.582576 | tf(freq=21.0), with freq of:
                            21.0 | termFreq=21.0
                        6.976686 | idf(docFreq=9021, maxDocs=3555860)
                        0.046875 | fieldNorm(doc=77874)
        0.78441995 | min of:
            0.78441995 | function score, score mode [multiply]
                0.7923434 | function score, product of:
                    1.0 | match filter: *:*
```

https://explain.commonsearch.org/?q=python&g=en

Issues with this architecture

- Static & dynamic scoring are in different codebases
- No control over result diversity
- Hard to optimize
- Very dependent on Elasticsearch

Rescoring


Indexer
words, static score, features
Database
top 1k docs, features
query
Rescorer
final 10 docs
Searcher
Users

Issues with rescoring

- Latency
- Pagination
- Harder to explain

Learning to rank

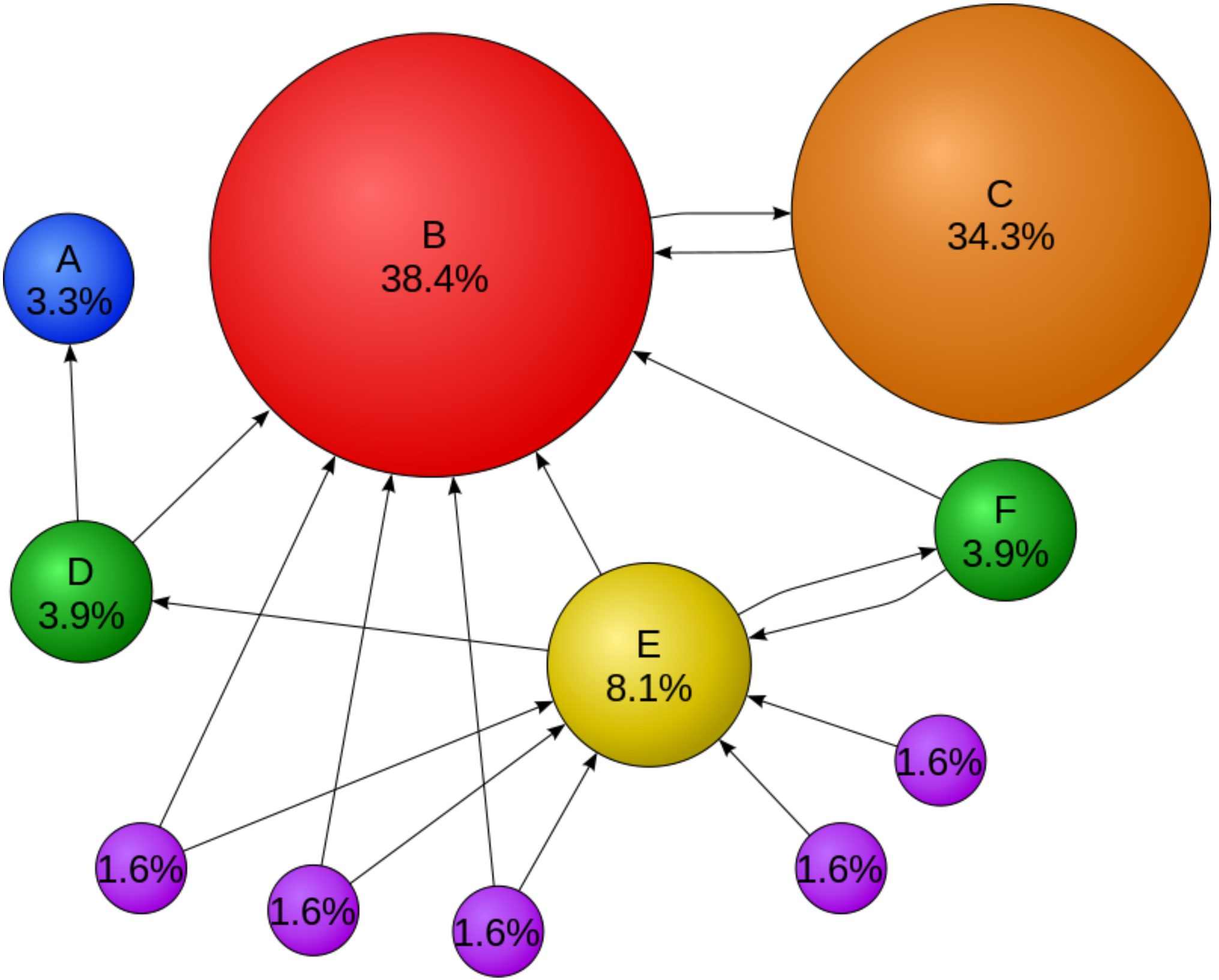
LTR Model

- Features
- Training dataset
- Evaluation: NDCG, ERR, ...
- Algorithms: AdaRank, ListNet, LambdaMART, ...
- Learning with Spark!

The right questions

- What do users expect?
- What features?
- How to evaluate and fine-tune in the real world?

PageRank with Spark


A
3.3%
B
38.4%
C
34.3%
D
3.9%
E
8.1%
F
3.9%
1.6%
1.6%
1.6%
1.6%
1.6%

October 2016 Crawl Archive Now Available

November 7, 2016 **Sebastian Nagel**

The crawl archive for October 2016 is now available! The archive is located in the **commoncrawl** bucket at **crawl-data/CC-MAIN-2016-44/.** It contains more than 3.25 billion web pages.

Similar to the September crawl, we used sitemaps to improve the crawl seed list, including sitemaps named in the robots.txt file of the top-million domains from Alexa, and sitemaps from the top 150,000 hosts in Common Search's host-level page ranks. The maximum number of URL's extracted per domain was 200,000. The resulting crawl included 2 billion new URLs, not contained in previous crawls.

We are grateful to webxtrakt for donating a list of 14 million verified, DNS-resolvable domain names of European country-code TLDs (eu, .fr, .be, .de, .ch, .nl, .pl). We included these domains into the October crawl and we hope for a ongoing partnership with webxtract to improve the coverage of the crawls.

To assist with exploring and using the dataset, we provide gzipped files that list:

- all segments (CC-MAIN-2016-44/segment.paths.gz)
- all WARC files (CC-MAIN-2016-44/warc.paths.gz)
- all WAT files (CC-MAIN-2016-44/wat.paths.gz)
- all WET files (CC-MAIN-2016-44/wet.paths.gz)
- robots.txt files (CC-MAIN-2016-44/robotstxt.paths.gz)
- non-200 HTTP status code responses (CC-MAIN-2016-44/non200responses.paths.gz)

Recent Posts

October 2016 Crawl Archive Now Available

September 2016 Crawl Archive Now Available

News Dataset Available

August 2016 Crawl Archive Now Available

Data Sets Containing Robots.txt Files and Non-200 Responses

http://commoncrawl.org

commonsearch / cosr-back

Unwatch 17 | Unstar 94 | Fork 29

Code | Issues 41 | Pull requests 1 | Projects 0 | Pulse | Graphs | Settings

Backend of Common Search. Analyses webpages and sends them to the index. https://about.commonsearch.org — Edit

111 commits | 3 branches | 0 releases | 11 contributors | Apache-2.0

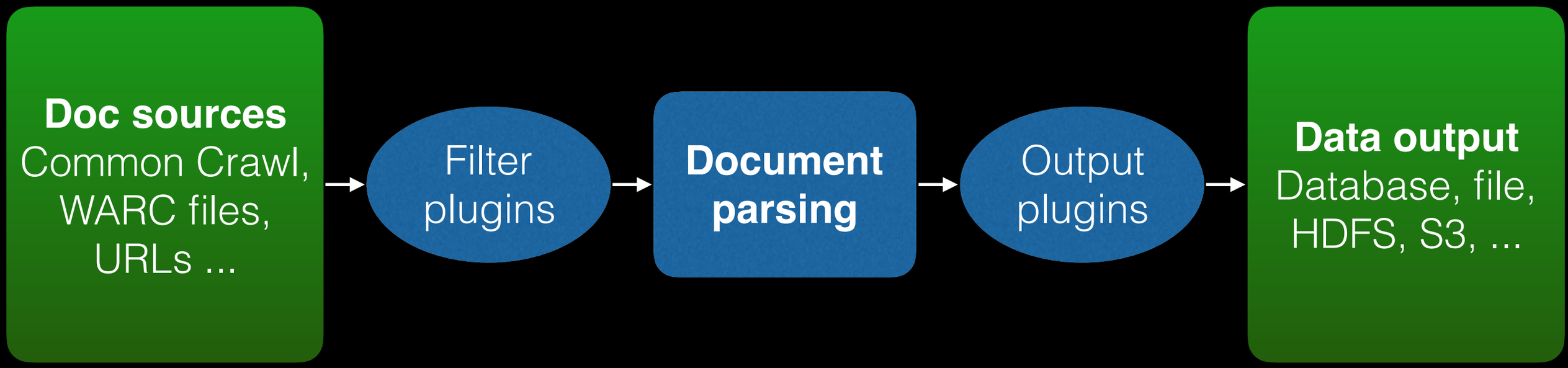
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

HenriqueLimas committed with sylvinus Add Commonsearch PageRank Signal (#70) ... Latest commit 6b44fee 14 days ago

cosrlib	Add Commonsearch PageRank Signal (#70)	14 days ago
explainer	Initial Python 3 prep work	2 months ago
plugins	PageRank fixes for upcoming tutorial	2 months ago
scripts	Scale back on the use of DataFrames because of Java Heap issues	3 months ago
spark	Bug correction #62 adding option --overwrite for pagerank job. (#69)	15 days ago
tests	Add Commonsearch PageRank Signal (#70)	14 days ago
urlserver	Add Commonsearch PageRank Signal (#70)	14 days ago
.coveragerc	Add coverage via coveralls for #32	8 months ago
.dockerhash	Upgrade Protobug (wasn't building properly anymore)	2 months ago
.dockerignore	Inspect docker images on Travis	4 months ago
.gitignore	Upgrade to Spark 2.0.0 and refactor WebGraph + PageRank jobs to use S...	3 months ago

https://github.com/commonsearch/cosr-back

Common Search Pipeline




```
spark-submit [spark_options] \
    /cosr/back/spark/jobs/pipeline.py \
    --source [source_options] \
    --plugin [plugin_options] \
    [other_pipeline_options]
```

Most popular Wikipedia pages

```
spark-submit --verbose \
    /cosr/back/spark/jobs/pipeline.py \
    --source commoncrawl:limit=8,maxdocs=1000 \
    --plugin "plugins.filter.Domains:skip=1,domains=tumblr.com wordpress.com" \
    --plugin plugins.backlinks.MostExternallyLinkedPages:domain=wikipedia.org,path=out/top_wikipedia/ \
    --stop_delay 600
```

Dumping the web graph

```
spark-submit --verbose \
    /cosr/back/spark/jobs/pipeline.py \
--source commoncrawl:limit=4,maxdocs=100 \
--plugin plugins.webgraph.DomainToDomainParquet:path=out/webgraph/
```

Naive pyspark PageRank

```
from operator import add

def compute_contribs(urls, rank):
    """Calculates URL contributions to the rank of other URLs."""
    num_urls = len(urls)
    for url in urls:
        yield (url, rank / num_urls)

labels = sqlc.read.load(os.path.join(self.args.webgraph, "vertices")).rdd
lines = sqlc.read.load(os.path.join(self.args.webgraph, "edges")).rdd

# Loads all URLs from input file and initialize their neighbors.
links = lines.map(lambda row: (row.src, row.dst)).distinct().groupByKey().mapValues(list).cache()

# Loads all URLs with other URL(s) link to from input file and initialize ranks of them to one.
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))

# Calculates and updates URL ranks continuously using PageRank algorithm.
for iteration in range(self.args.maxiter):

    # Calculates URL contributions to the rank of other URLs.
    contribs = links.join(ranks).flatMap(
        lambda url_urls_rank: compute_contribs(url_urls_rank[1][0], url_urls_rank[1][1]))

    # Re-calculates URL ranks based on neighbor contributions.
    ranks = contribs.reduceByKey(add).mapValues(lambda rank: rank * 0.85 + 0.15)

# Restores the labels from the vertices file
labelled_ranks = labels.leftOuterJoin(ranks).map(
    lambda row: "%s %s" % (row[1][0], row[1][1] or 0.15)
)

if self.args.output:

    labelled_ranks.coalesce(1).saveAsTextFile(
        self.args.output
    )
```

GraphFrames

```
from graphframes import GraphFrame  # pylint: disable=import-error

edge_df = sqlc.read.load(os.path.join(self.args.webgraph, "edges"))
vertex_df = sqlc.read.load(os.path.join(self.args.webgraph, "vertices"))

graph = GraphFrame(vertex_df, edge_df)

ranked_graph = graph.pageRank(maxIter=self.args.maxiter)

final_df = sql(sqlc, """
    SELECT CONCAT(ranks.domain, ' ', ranks.pagerank) r
    FROM ranks
    ORDER BY ranks.pagerank DESC
""", {"ranks": ranked_graph.vertices})

if self.args.output:

    final_df.coalesce(1).write.text(
        self.args.output,
        compression="gzip" if self.args.gzip else "none"
    )
```

SparkSQL PageRank

```
for iteration in range(self.args.maxiter):

    changed_ranks_df = sql(sqlc, """
        SELECT
            edges.dst id,
            cast(
                0.15 + 0.85 * sum(COALESCE(ranks_src.rank, 0.15) * edges.weight)
                as float
            ) rank_new,
            first(ranks_dst.rank) rank_old
        FROM edges
        LEFT OUTER JOIN ranks_src ON edges.src = ranks_src.id
        LEFT OUTER JOIN ranks_dst ON edges.dst = ranks_dst.id
        GROUP BY edges.dst
        HAVING ABS(rank_old - rank_new) > %s
    """ % self.args.precision, {"ranks_src": ranks_df, "ranks_dst": ranks_df, "edges": edge_df})
```

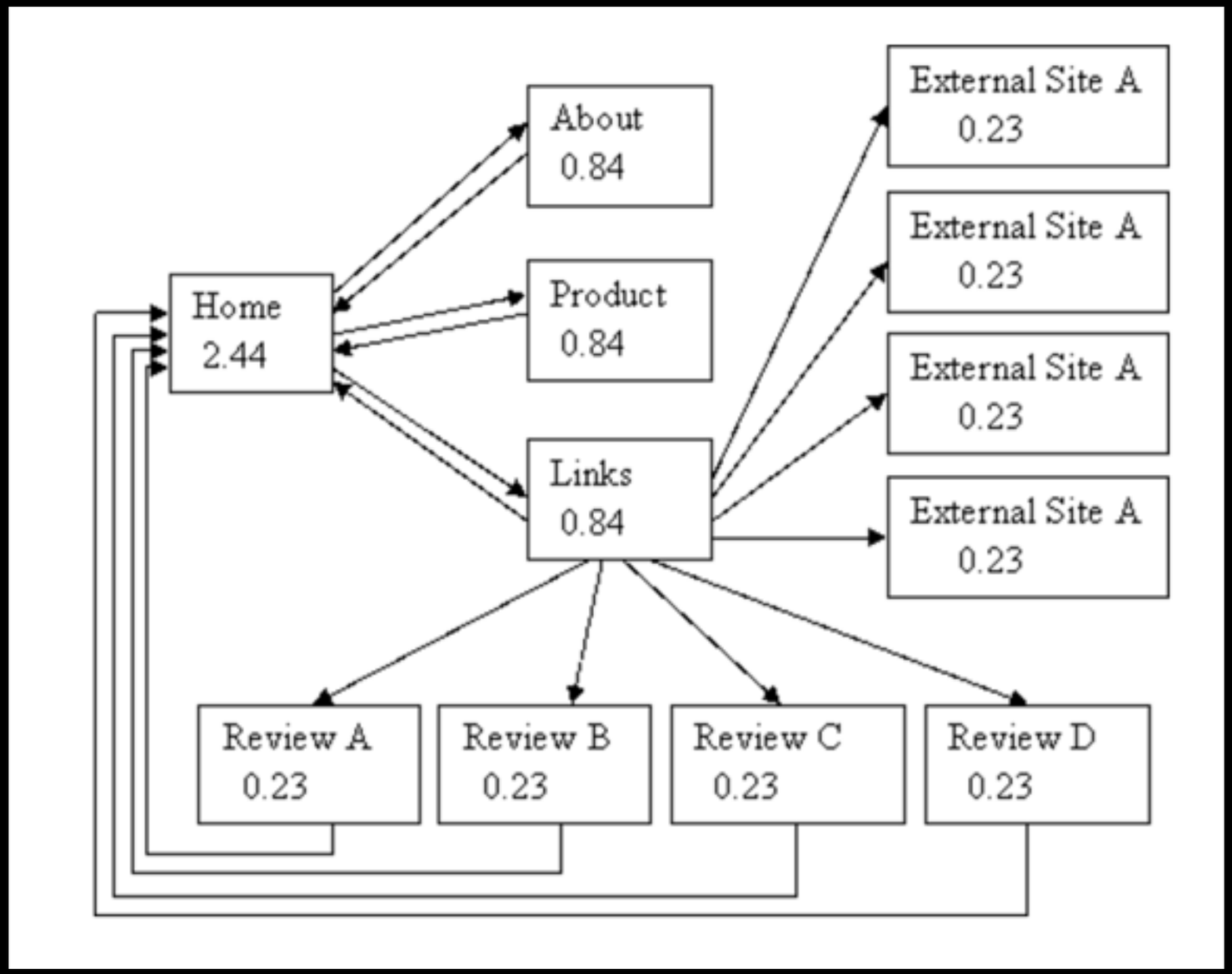
```
new_ranks_df = sql(sqlc, """
    SELECT ranks.id id, COALESCE(changed_ranks.rank_new, ranks.rank) rank
    FROM ranks
    LEFT JOIN changed_ranks ON changed_ranks.id = ranks.id
""", {"ranks": ranks_df, "changed_ranks": changed_ranks_df})
```

SparkSQL PageRank

```
stats_df = sql(sqlc, """
    SELECT
        sum(abs(rank_new - rank_old)) as sum_diff,
        count(*) as count_diff,
        min(abs(rank_new - rank_old)) as min_diff,
        max(abs(rank_new - rank_old)) as max_diff,
        avg(abs(rank_new - rank_old)) as avg_diff,
        stddev(abs(rank_new - rank_old)) as stddev_diff
    FROM changes
""", {"changes": changed_ranks_df})
```

https://github.com/commonsearch/cosr-back/blob/master/spark/jobs/pagerank.py

Tests



http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm

https://github.com/commonsearch/cosr-back/blob/master/tests/sparktests/test_pagerank.py

Tutorial: Running PageRank on the Web

This tutorial get you through all the steps required to run PageRank on billions of pages using Common Search's codebase and tools such as Apache Spark and AWS.

1. Prerequisites

You should go through our Analyzing the web with Spark on EC2 first, to install the required software, understand the basic concepts of our pipeline, and run a simpler job first, at least on your local machine.

You should also be familiar with basic Graph theory.

2. Dumping the Web Graph

Before computing PageRank, we need to parse all the link in our corpus and save them as a directed graph.

(In some cases, you can actually skip this step by using one of the dumps we publish directly.)

To dump the web graph, we are doing to use the `webgraph` plugin. Here is how you would dump it for the first 400 URLs from Common Crawl, at the host level:

```
spark-submit --verbose \
    /cosr/back/spark/jobs/pipeline.py \
    --source commoncrawl:limit=4,maxdocs=100 \
    --plugin plugins.webgraph.DomainToDomainParquet:path=out/webgraph/ \
    --stop_delay 600
```

This will actually create 2 subdirectories in `out/webgraph/`: one for the vertices and one for the edges. Both dumps will be stored as Apache Parquet format, so that we can easily reuse them in the next step.

You might notice this command will go over the source documents multiple times. This shouldn't be a big issue with so few

https://about.commonsearch.org/developer/get-started

Our first public datasets: Host-level WebGraph and PageRank!

Jul 31, 2016 Back to blog

Common Search is building an open source search engine with transparent rankings, and analyzing the hyperlinks on the web is a major part of this effort.

To make that possible, we are going to publish datasets that will let contributors, students and researchers reproduce the rankings, submit improvements and hopefully use the underlying data for their own work.

The first two we are happy to announce today are a **host-level WebGraph** and a list of **host-level PageRanks**.

We want to give credit to both Common Crawl for their amazing work and to the Web Data Commons project who published similar dumps in 2012 and 2014.

Host-level WebGraph

This dataset is based on the June 2016 Common Crawl. It represents the directed graph of all hyperlinks aggregated at the hostname level (e.g. "about.commonsearch.org").

- vertices.txt.gz (575M lines, 2.3 GB). Format: `[int64 id] [hostname]`
- edges.txt.gz (112M lines, 4.7 GB). Format: `[int64 src_id] [int64 dst_id]`

The Python code used to generate these files is available on GitHub!

Host-level PageRank

This dataset was generated directly from the Host-level WebGraph above and contains a PageRank for

Top 10

```
facebook.com 244660.58
twitter.com 164232.66
blogger.com 77521.93
youtube.com 62967.95
plus.google.com 61344.234
instagram.com 39883.676
linkedin.com 34856.848
wordpress.org 33809.844
google.com 27425.883
pinterest.com 25640.172
```

```r
#Grab the data
df <- read.csv("pagerank-top1m.txt", header = F, sep = " ")

#Log Normalize
logNorm <- function(x){

    #Normalize
    x <- (x-min(x))/(max(x)-min(x))
    10 / (1 - (log10(x)*.25))
}

#Append a Column named PR to the dataset
df$pr <- (round(logNorm(df$V2),digits = 0))
```

Filter

	V1	V2	pr
1	facebook.com	244660.580	10
2	twitter.com	164232.660	10
3	blogger.com	77521.930	9
4	youtube.com	62967.950	9
5	plus.google.com	61344.234	9
6	instagram.com	39883.676	9
7	linkedin.com	34856.848	9
8	wordpress.org	33809.844	9
9	google.com	27425.883	8
10	pinterest.com	25640.172	8
11	blog.instagram.com	20564.566	8
12	irisvisia.com	20118.588	8
13	myfunfan.com	17735.658	8
14	adobe.com	17735.658	8
15	help.instagram.com	16960.008	8
16	networkadvertising.org	16117.567	8
17	4.bp.blogspot.com	15581.531	8

Spam

Hot News

About 1,520,000 results (0.39 seconds)

Ads by Google

Sell Your Settlement
www.cashforannuity.org/ (877) 999-0457
Highest Payouts. Fastest Payouts. Instantly Get a Fast, Free Quote.
Highest Payout Guarantee · Free Expert Consultation · Fast Payouts

Structured Settlements
www.sellmyannuity.net/
Cash in Your **Structured Settlement** The #1 Online Resource
Call Us Today · 3 Steps to Get Cash Now

Structured Settlement Cash Now - Get Your Free Quote Right Now.
cash.stonestreet.com/**Settlement**_Cash/Free_Quote (866) 390-0583
Sell Some or All Your **Structured Settlement**. Up-Front Cash. Top Dollar Payouts.
Confidential · 25+ Years in Business · Advances Available · Cash in 24 Hours
Customer Reviews - Cash Now for Annuities

Structured Settlement Funding
www.pravatilegalfunding.com/ (844) 454-4892
Get Your Money Before You **Settle**. Keep The Money If You Don't Win!
Get Your Money Now · Low Legal Funding Rates · Live Chat 24/7 · Free Case Review
A+ BBB Rating, Accredited Business - Better Business Bureau
Contact Us · Funding Services · Plaintiff Acquisition · Who We Are · Legal Funding

Galery C
Joker
November
Vikings
Standing Rock
Where Do I Vo
Donna Brazile
Jamie Collins
Dia de los Mue
Heidi Klum
Chipotle
Trick or treat 2
SEPTA strike
Jalen Hurd
Dancing with t
Warriors
Val Kilmer
Andre Johnson

MP3GP.XYZ

Search for music...

USA · COUNTRY · KIDS · ROCK · U.K · J-POP · LATIN · YOUTUBE

Home / Music Videos / teri yaad ka amrit mahendra kapoor

teri yaad ka amrit mahendra kapoor

TERI Yaad Ka Amrit - Mahendra Kapoor - True Yog - BK Meditat
An Excellent Meditation Song from Brahma Kumaris. The True Unadu
Knowledgeful Remembrance of Baba-Supreme
Download . Play

TUMHARA Pyar O Baba - Mahendra Kapoor ji - Great BK Medita
13/108.
19-07-2016: Tumhara Pyar O Baba... Lyrics & Translation for this divi
meditation song by
Download . Play

Suresh Wadkar-Teri Yaad Ka Deep-Great Mesditation Song Fro
Kumaris.
The True Unselfish Remembrance Of Baba-Supreme Father-Shiv-Alla
to the Door To Heaven,
Download . Play

aktiendepot.com

Trading and strategy games steam

Die Erkenntnis, dass der Handel mit Hailed as the original FPS-RPG game that spawned countless imitators. Stride: Veteran Edition is a love letter to Steam Achievements. Steam Trading Cards.12. Febr 2015 "A surprisingly fun trading / economy game set on Mars. Offworld Trading Company is a real-time strategy game in which money, not high probability day trading strategies and systems pdfCategory: Single-player, Steam Achievements, Full controller support, Steam Trading Cards, Steam Cloud Release date: May 20, 2014. Price: $19.99trendline trading strategy pdf6. Mai 2014 Steam VPN Aktivierung - Ganz einfach mit der schnellen Anlaufung von - deinem uncut Games Shop aus Österreich. Final Fantasy Trading Card Game Online. Party. Puzzle. Rennspiel. Sammlung. Simulation. Sport. Strategie. ——— Hardware. Film. Buch. Merchandise etc Monaco – What's Yours Is Mine - Klöher de Klöher de cfd trading tricks29. Jan. 2015 17.02.2015 um 16:46 Uhr Stardock hat ein neues Echtzeit-Strategiespiel angekündigt, das sich nicht um Schlachten, sondern um Handel und 5. Aug. 2013 Seit der Einführung der Steam Trading Cards können sich Spieler neben Badges, Peter Grubmair ist Redakteur bei Gamezone und das letzte Ungeheuer aus ... AR-K im Lasertest. Bitte nicht noch ein Kickstarter-Adventure!

Offworld Trading Company kaufen, OTC Key - MMOGA

Valve: Delivering Content to More Than 125 Million Users kfz handel coesfeld 7. Juni 2016 Schweizer Strategie-Game wird mit Gold überhäuft ... wir im Sommer eine Early-Access-Version auf Steam stellen und das fertige Game dann 29. März 2016 Offworld Trading Company: Mehr als genug Freiraum! Das knallharte Strategie-Game Offworld Trading Company von 2015 will[] keine prozedurale Umgebung für Games aus

Den richtigen Broker finden:
Computerhandel potsdam
Investitionsfinanzierung i bank lexington
Trading strategy pdf zusammenfügen
Wotan werkzeughandel e.k

"Gehen Sie an die Börse, und stecken Sie Ihr Geld in Aktien. Dann kaufen Sie sich in einer Apotheke eine große Dosis Schlaftabletten. Nach ein Jahren wachen Sie als reicher Mann auf."
André Kostolany

Home

ngc wallpaper | Home · Popular · New · Random · Browse Category

ngc wallpaper

National Geographic provides free desktop wallpapers of animals, travel, nature, weather, underwater, adventure, exploration, people, culture, science, space, weather photos, and more

Enter search phrase... i.e tiger, animal water, etc.

Check Out T
Offer!
Try Now !

Random Wallpaper

Hipster Floral Fresh New Wallpaper 1366x768 Full HD Wallpapers

New Wallpapers

Wallpaper Underlayment bd wallfon 3

All of About Hindi Musics

The World Musics, Hindi Musics

HOME

Type the song title or singer...

Gulshan Kumar All Song Bhakti Pagalworld

Home » Music » Gulshan Kumar All Song Bhakti Pagalworld

musicinctours - **Gulshan Kumar All Song Bhakti Pagalworld, 2.91 MB** MP3 Download, Before playing or download any content related with Gulshan Kumar All Song Bhakti Pagalworld below, you must agree with our Terms of Service. If You are the owner of contents are displayed on our site, please inform me with the URL of copyrighting content by fill DMCA Contact Form for removal request. Will be processed in 3 business days.

Hanuman Chalisa with Subtitles [Full Song] Gulshan Kumar, Hariharan - Shree Hanuman Chalisa
VIDEO · DOWNLOAD

Aa Maa Aa Tujhe Dil Ne Pukara Gulshan Kumar [Full Song] Mamta Ka Mandir
VIDEO · DOWNLOAD

Bhor Bhai Din Devi Bhajan By Gulshan Kumar

RECENT
- 50 To
- Bollyv

Spamdexing

- Keyword stuffing, hidden text
- Scraper sites, Mirrors
- Link farms
- Splogs, Comment spam
- Domaining
- Cloaking
- Bombing

Questions?

https://about.commonsearch.org/contributing

https://github.com/commonsearch

contact@commonsearch.org

slack.commonsearch.org