DIPLOMACIA

Marcelo ficou “muito impressionado com a personalidade política” de Modi

O Presidente da República está de visita de estado à Índia.

Lusa - 15 de Fevereiro de 2020, 11:43

O Presidente da República, Marcelo Rebelo de Sousa, declarou neste sábado ter ficado “muito impressionado com a personalidade política” do primeiro-ministro indiano, Narendra Modi, e com o seu empenho no reforço das relações luso-indianas.
DIPLOMACIA

Marcelo ficou “muito impressionado com a personalidade política” de Modi

O Presidente da República está de visita de estado à Índia.

Lusa - 15 de Fevereiro de 2020, 13:43

O Presidente da República, Marcelo Rebelo de Sousa, declarou neste sábado ter ficado “muito impressionado com a personalidade política” do primeiro-ministro indiano, Narendra Modi, e com o seu empenho no reforço das relações luso-indianas.
DIPLOMACIA

Marcelo ficou “muito impressionado com a personalidade política” de Modi

O Presidente da República está de visita de estado à Índia.

Lusa  ·  15 de Fevereiro de 2020, 13:43

O Presidente da República, Marcelo Rebelo de Sousa, declarou neste sábado ter ficado “muito impressionado com a personalidade política” do primeiro-ministro indiano, Narendra Modi, e com o seu empenho no reforço das relações luso-indianas.
~50% of the page is made of images
Why does image search matter?

sparktoro.com/blog/new-jumpshot-2018-data-where-searches-happen-on-the-web-google-amazon-facebook-beyond/
Why does image search matter?

1 in 4 web searches is for images

1 in 4 web searches is for images
What about Arquivo.pt?

Unique pageviews per service: 2020/04/20 - 2020/05/20

- Wayback: 89,489 views (89.0%)
- Image Search: 89,18 views (8.9%)
- Page Search: 2,136 views (2.1%)

Arquivo.pt Google Analytics
What about Arquivo.pt?

1 in 5 Arquivo.pt searches is for images
Arquivo.pt Image Search
Arquivo.pt APIs

● Arquivo.pt makes **8,000+ million pages** and **1,800+ million images** available for visualization and search:
  ○ Archived web pages -> **Text Search API**/Memento/CDX Server
  ○ Text and metadata search -> **Text Search API**
  ○ Image search -> **Image Search API**

● Available to the general public without registration
● Open Source

● [https://github.com/arquivo/pwa-technologies/wiki/APIs](https://github.com/arquivo/pwa-technologies/wiki/APIs)
Arquivo.pt Image Search (as of Jan 2020)

<table>
<thead>
<tr>
<th>Category</th>
<th>Count/Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexed images</td>
<td>22 million</td>
</tr>
<tr>
<td>Collection count</td>
<td>90</td>
</tr>
<tr>
<td>(W)ARCs</td>
<td>3 million</td>
</tr>
<tr>
<td>(W)ARC sizes</td>
<td>334 TB</td>
</tr>
<tr>
<td>Total collected files</td>
<td>6,000 million</td>
</tr>
<tr>
<td>Total collected images</td>
<td>1,602 million</td>
</tr>
<tr>
<td>Oldest image date</td>
<td>15/04/1994</td>
</tr>
<tr>
<td>Newest image date</td>
<td>14/11/2019</td>
</tr>
</tbody>
</table>
## Arquivo.pt Image Search (as of Jan 2020)

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexed images</td>
<td>22 million</td>
</tr>
<tr>
<td>Collection count</td>
<td>90</td>
</tr>
<tr>
<td>(W)ARCs</td>
<td>3 million</td>
</tr>
<tr>
<td>(W)ARC sizes</td>
<td>334 TB</td>
</tr>
<tr>
<td>Total collected files</td>
<td>6,000 million</td>
</tr>
<tr>
<td>Total collected images</td>
<td>1,602 million</td>
</tr>
<tr>
<td>Oldest image date</td>
<td>15/04/1994</td>
</tr>
<tr>
<td>Newest image date</td>
<td>14/11/2019</td>
</tr>
</tbody>
</table>
Arquivo.pt Image Search (as of March 2021)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexed images</td>
<td>1,862 million</td>
</tr>
<tr>
<td>Unique images</td>
<td>584 million</td>
</tr>
<tr>
<td>Collection count</td>
<td>115</td>
</tr>
<tr>
<td>(W)ARCs</td>
<td>5 million</td>
</tr>
<tr>
<td>(W)ARC sizes</td>
<td>520 TB</td>
</tr>
<tr>
<td>Total collected files</td>
<td>8,500 million</td>
</tr>
<tr>
<td>Total collected images</td>
<td>2,408 million</td>
</tr>
<tr>
<td>Oldest image date</td>
<td>15/04/1994</td>
</tr>
<tr>
<td>Newest image date</td>
<td>14/11/2020</td>
</tr>
</tbody>
</table>
Opportunities for improvement

- Lack of image specific metadata
  - 43% (10,163,080 images) without imgAlt or imgTitle
- Why is the difference between collected and indexed so large?
- Only the oldest page per image is indexed
- Search result ranking does not take image popularity into account
Potential solutions

● Index all* pages that mention an image
● Solve relative URL issues and find images in more places on page
● Remove MongoDB and use only Hadoop/HDFS

● Extract image caption from text surrounding images

● Use correct Solr types
● Extract metadata and use it for ranking
  ○ Number of times an image appear on page; number of times its metadata changes
From images to metadata

- Image search is only as good as the associated metadata
- If we only look into the (W)ARC image records, we only have information about the image URL and image date, which is not very informative
- Where can we find this information?
From images to metadata

- Image search is only as good as the associated metadata
- If we only look into the (W)ARC image records, we only have information about the image URL and image date, which is not very informative
- Where can we find this information?
- HTML PAGES!
The anatomy of a webpage

News regarding Marcelo’s visit to India

- Main image:
  - Marcelo Rebelo de Sousa, President of Portugal giving a speech as a part of the state visit to India
  - Caption: “Marcelo Rebelo de Sousa LUSA/ESTELA SILVA”

- Secondary images:
  - Person/Author of opinion piece
  - Soccer practice “stock photo”
  - Renovation of an apartment
  - Social network share icons
  - Público’s Logo
  - Other navigational buttons

- Links to external images
- Images as a CSS background

[Link to the article](publico.pt/2020/02/15/politica/noticia/marcelo-ficou-impressionado-personalidade-politica-modi-1904277)
Inside the skeleton
Finding images in pages

- `<img>` tag attributes
- `<a>` tag attributes
- Inline CSS background images
- Inline base64 images
- Images set by JS
- `<figure>`, `<picture>`
Takeways for finding images in pages

- **<img> tag attributes**
  - everything in `src` (regardless of image extension)
  - other attributes that match list of image extensions

- **<a> tag attributes**
  - `href` that match list of image extensions

- **Inline CSS background images**
  - `background-url:` that match list of image extensions

- **Base64 images**

- **Fixed relative URL solver**
Finding images in pages results

- `<img>` tag attributes
- `<a>` tag attributes
- Inline CSS background images
- Inline base64 images
- Images set by JS
- `<figure>`, `<picture>`

<table>
<thead>
<tr>
<th>Percentage of references</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&lt;img&gt;</strong></td>
</tr>
<tr>
<td>90.6%</td>
</tr>
<tr>
<td><strong>&lt;a&gt;</strong></td>
</tr>
<tr>
<td>8.7%</td>
</tr>
<tr>
<td><strong>css</strong></td>
</tr>
<tr>
<td>0.7%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Percentage of references</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Normal images</strong></td>
</tr>
<tr>
<td>99.9%</td>
</tr>
<tr>
<td><strong>base64</strong></td>
</tr>
<tr>
<td>0.1%</td>
</tr>
</tbody>
</table>
From page metadata to image metadata

The following attributes are common to all images that show up in a page:

- **Page Title**
  - Page title attribute; it is used to provide additional information about an HTML page;

- **Page URL Tokens**
  - The keywords of the URL of the HTML page that contains the image.

But this general information may not be relevant to all images.
Matching images to HTML <img> tags

pageTitle="Marcelo ficou “muito impressionado com a personalidade política” de Modi | Diplomacia | PÚBLICO"
pageSrcTokens="https www publico pt 2020 02 15 politica noticia marcelo ficou (...)"

pageTitle="Marcelo ficou “muito impressionado com a personalidade política” de Modi | Diplomacia | PÚBLICO"
pageSrcTokens="https www publico pt 2020 02 15 politica noticia marcelo ficou (...)"

pageTitle="Marcelo ficou “muito impressionado com a personalidade política” de Modi | Diplomacia | PÚBLICO"
pageSrcTokens="https www publico pt 2020 02 15 politica noticia marcelo ficou (...)"
Metadata: `<img>` tag attributes

We select all `<img>` tags in the html and extract the following metadata:

- **imgSrcTokens**
  - an image by a URL, which often includes the filename of the image
- **imgTitle**
  - it provides additional information about the image;
- **imgAlt**
  - it provides alternative information about an image if a user cannot view it;
pageTitle="Marcelo ficou “muito impressionado com a personalidade política” de Modi | Diplomacia | PÚBLICO"
pageSrcTokens="https www publico pt 2020 02 15 politica noticia marcelo ficou (...)"

pageTitle="Marcelo ficou “muito impressionado com a personalidade política” de Modi | Diplomacia | PÚBLICO"
pageSrcTokens="https www publico pt 2020 02 15 politica noticia marcelo ficou (...)"

pageTitle="Marcelo ficou “muito impressionado com a personalidade política” de Modi | Diplomacia | PÚBLICO"
pageSrcTokens="https www publico pt 2020 02 15 politica noticia marcelo ficou (...)"
Metadata: <img> tag attributes

We select all <img> tags in the html and extract the following metadata:

- **imgSrcTokens**
  - an image by a URL, which often includes the filename of the image
- **imgTitle**
  - it provides additional information about the image;
- **imgAlt**
  - it provides alternative information about an image if a user cannot view it;
- **imgCaption**
  - portion of the HTML page text that is closest to the image
Finding an image caption

Finding an image caption


Image caption extraction

I arrived at the following method

First parent with text
- Default method
- Works well for images in boxes or *reasonably* structured pages
Image caption extraction

I arrived at the following method

First parent with text
- Default method
- Works well for images in boxes or reasonably structured pages

Previous and next node text
- Used if the first parent with text is at the level of the page with more siblings
- List of images as in a blog
I arrived at the following method

First parent with text
- Default method
- Works well for images in boxes or reasonably structured pages

Previous and next node text
- Used if the first parent with text is at the level of the page with more siblings
- List of images as in a blog
Associar palavras do elemento *parent* do HTML à imagem (legendas)

Texto do *parent* *funciona* em páginas com HTML correctamente estruturado

Blog do futebol

*Futebol*

Cristiano Ronaldo eleito melhor jogador do mundo

*Messi devastated with second place*

*Messi de rastos com o segundo lugar*
Hipótese falha em páginas com estrutura “flat”

Texto do elemento *parent* falha em páginas mal estruturadas (sem separação entre tipos de conteúdo semântico)

Blog do futebol

**Futebol**

Cristiano Ronaldo eleito melhor jogador do mundo

**Futebol**

Messi de rastos com o segundo lugar
Solução método híbrido: `parent.text()` OR `sibling.text()`

- **Páginas normais:** `text` do `parent`
- **Páginas com estrutura flat:** `text` dos nós adjacentes (`siblings`)

Blog do futebol

**Futebol**

Cristiano Ronaldo eleito melhor jogador do mundo

**Futebol**

Messi devastado com o segundo lugar
Matching images to HTML `<img>` tags

pageTitle="Marcelo ficou "muito impressionado com a personalidade política" de Modi (...)"
pageSrcTokens="(...)"
marcelo ficou (...)"

pageTitle="Marcelo ficou "muito impressionado (...)"
pageSrcTokens="(...)"

pageTitle="Marcelo ficou "muito impressionado (...)"
pageSrcTokens="(...)"

imgAlt="Marcelo Rebelo de Sousa"
imgTitle=""
imgSrcTokens="image ns publico pt imagens aspx (...)"

imgCaption="Marcelo Rebelo de Sousa LUSA/ESTELA SILVA"

imgAlt=""
imgTitle=""
imgSrcTokens="image ns publico pt imagens aspx (...)"

imgCaption="FUTEBOL Moussa Marega, deixa-me dizer-te uma coisa - Opinião de Adriano Miranda"

imgAlt=""
imgTitle=""
imgSrcTokens="image ns publico pt imagens aspx (...)"

imgCaption="ARQUITECTURA A renovação deste apartamento é uma viagem à Lisboa do passado"
An alternative way to find images on the page is find direct links to images.

To do this, we select links (\(<a>\)) that point to files with image extensions, and extract the following metadata:

- `imgSrcTokens`
- `imgFilename`
- `imgCaption` ({\(<a>\)} anchor text)
  - `The text inside the link is used as the image caption`
An increasingly popular way of placing images on the web is through the use of the CSS background attribute, which places the image inside an HTML element (usually a div)

These images are referenced through a CSS background:url('<url>')

- imgSrcTokens
- imgFilename
Image metadata takeways

- Extracted image caption for images found in `<img>`
- Used anchor text as image caption for images found in `<a>`
- Used only page metadata for css images
Metadata Open Questions

How many images have explicit metadata?
- Current measurements show over 99% of the images have metadata (imgCaption)

But how can we measure the quality of this metadata?
- IMG_00123.jpg is not a very helpful entry

What to do when this metadata is missing (CSS or orphan image records) or wrong?
- Page title, top-k terms....
- Deep Image Captioning/Classification techniques
Examine the quality of the metadata
Encodings on the internet

“Lote para construção de moradia Inclui projecto a aprovar Ârea do lote --Â» 360 m2 Com frente de 20 metros”

“Espectáculos a não perder 09/04/2009 | Sem Comentários | Concertos”

“Mobidogs Sempre sonhou ter um cÃ£o, um companheiro simpÃ¡tico que esteja ao seu lado para partilhar as suas alegrias e desgostos.... 4.00EUR”
public static String decode(byte[] arcRecordBytes) throws IOException {
    String recordEncoding = ImageSearchIndexingUtil.guessEncoding(arcRecordBytes);
    InputStream is = new ByteArrayInputStream(arcRecordBytes);
    String html = IOUtils.toString(is, recordEncoding);
    //if chars in UTF8_MISMATCH were detected, the page is in UTF_8 but encoded in ISO_8859_1
    //if we re-encode the string, the accented chars will be correctly represented
    if (ImageSearchIndexingUtil.UTF8_MISMATCH.matcher(html).find()) {
        byte[] b = html.getBytes(StandardCharsets.ISO_8859_1);
        String newHtml = new String(b, StandardCharsets.UTF_8);
        //if the chars are detected again, the page is beyond repair and the initial encoding is used
        if (!ImageSearchIndexingUtil.UTF8_MISMATCH.matcher(newHtml).find()) {
            html = newHtml;
        }
    }

    return html;
}
public static String decode(byte[] arcRecordBytes) throws IOException {
    String recordEncoding = ImageSearchIndexingUtil.guessEncoding(arcRecordBytes);
    InputStream is = new ByteArrayInputStream(arcRecordBytes);
    String html = IOUtils.toString(is, recordEncoding);
    //if chars in UTF8_MISMATCH were detected, the page is in UTF_8 but encoded in ISO_8859_1
    //if we re-encode the string, the accented chars will be correctly represented
    if (ImageSearchIndexingUtil.UTF8_MISMATCH.matcher(html).find()){
        byte[] b = html.getBytes(StandardCharsets.ISO_8859_1);
        String newHtml = new String(b, StandardCharsets.UTF_8);
        //if the chars are detected again, the page is beyond repair and the initial encoding is used
        if (!ImageSearchIndexingUtil.UTF8_MISMATCH.matcher(newHtml).find()){
            html = newHtml;
        }
    }
    return html;
}
“Lote para construção de moradia Incluí projecto a aprovar área do lote 360 m2 Com frente de 20 metros”

“Espectáculos a não perder 09/04/2009 | Sem Comentários | Concertos”

“Mobidogs Sempre sonhou ter um cão, um companheiro simpático que esteja ao seu lado para partilhar as suas alegrias e desgostos.... 4.00EUR”
Pages with wrong encoding in AWP4

Detector Mozilla: 160524
Detector Tika: 366202
Detector Mozilla + meu fix: 9665
But this does not solve all encoding issues


  amourao commented 15 days ago

  What is the URL that originated the issue?
  E.g.

  What happened?
  "เมียนมา" is being replaced by 🙁定点🙁定点🙁定点🙁定点

  What should have happened?
  HTML should have been parsed with correct encoding

- Won’t fix, only affects 1.4% of all extracted results, mostly in “exotic” encodings
Indexing Architecture
(W)ARC examples

https://imagens.publico.pt/...

https://imagens.publico.pt/imagens.aspx/1440184

https://imagens.publico.pt/imagens.aspx/1044361

<img alt="Marcelo Rebelo de Sousa" id="t4xiqy-interchange" (...)
src="https://imagens.publico.pt/imagens.aspx/1440184">

<img alt="" (...)
src="https://imagens.publico.pt/imagens.aspx/1044361">
(W)ARCs, HTML pages and images

During crawling, pages and images are added to a queue
Best case scenario, HTML and images are crawled together

- HTML and Images may be in different (W)ARCs
- HTML may reference non-crawled images
- Multiple versions of the image and HTML may be crawled
- URL may not be consistent across HTML images and img src
- Images may be referenced on more than one page
  - Same URL
  - Different URL, Same digest
  - Different URL, Different digest, near duplicate (e.g. resized versions)
  - Images may also change for the same url (Same URL, Different digest)
Old Map Reduce

**Phase 1: CreateImageDB**
- (1a) Find image records (i.e. records with mimetype that starts with image/)
- (1b) Create JSON record

**Phase 2: IndexImages**
- (2a) For all WARCS
- Find HTML records (i.e. records with mimetype that starts with text/html)
- Find all image tags in that html page
- (2b1) For each imgtag
  - search image in DB
  - (2b2) If image found
    - add new index for that image to the outputs.
- (2c) Delete non matching records

```
(1a) All WARCs
     /   \
(1b) POST Image URL (JSON)
     |      |
     v      v
Hadoop 3
     |      |
     v      v
MongoDB
     |      |
     v      v
(2a) All WARCs
     |      |
     v      v
Hadoop 3
```
Problems

- (W)ARCs are downloaded and parsed twice (images and HTML)
- Only the oldest page for each image is stored in the index
- MongoDB bottlenecks Hadoop parsing
- No logging is performed for what is happening
  - Failed (W)ARCs
  - Images parsed
  - ....
Solutions

- Parse HTML and image records in the same process
- Store all relevant pages for an image
- Rely on Hadoop/HDFS to match images to pages across WARC
  - Use image URL as Hadoop key
Extract images with metadata

Group by SURT
- For each record in the WARC
  - Find all image entries, extract metadata (height, width, img digest) and add them to the image SURT reduce list
  - Find all `<img>`/`<a>`/css entries in the HTML, extract metadata and add them to the image SURT reduce list

Merge metadata
- For each entry in the SURT, merge metadata for that record to produce a single record for each unique image (measured by digest)
  - This includes keeping `img title` and `alt` entries for images that show up in more than one page
  - Only add pages if they offer new image metadata (`title`, `alt` or `caption`)
Map Reduce: Extract images and metadata
How to deal with duplicate information?

- The amount of data produced by this step is huge!
- Generating a lot of documents for indexing
- But most of this information is duplicate
  - Images and pages that were crawled at different times but have not changed
  - References to the images that have the same caption/metadata
Deduplication potential solutions

- After careful examination, we arrived at the 3 deduplication scenarios:
  a. every page-image pair is a document
  b. the oldest page that references the image is the canonical document
  c. oldest page information and image specific information from all pages
     ■ keep reference to oldest page
     ■ Add all new image specific information (title, alt, caption) to the document
     ■ replace oldest page reference if a new oldest document shows up
What users want in image search?

- **Assumptions**
  - Users use page search to find pages
  - Users do not want to see duplicate images on the search results
  - Users do not use image search to find pages
  - No need to find all the pages that contain a given image
  - When an image appears on more than one page, finding the oldest page best matches the information need of a web archive user
    - Finding the page that better matches the image is not necessary
  - Technical details (imgAlt, ...) are rarely accessed by users

- **One Solr document per image with all page information**
  - Store page metadata for the oldest page
  - Store image specific metadata from all pages in a combined field
  - Remove fields that do not matter

- **Expected a decrease of 25-50% in index size**
Images in multiple pages

Indexed:
"id":"a4236137f5455edd8436a5d122c366fa62ba91139f45895f447565b3a6b926bb",
"imgSrc":"http://bp3.blogger.com/_v4YQruRZWLI/RxTDBM9FL5I/AAAAAAAALY/YJbzjdOKH0/s320/2.jpg",
"pageTstamp":"2008-02-15T08:40:21Z",
"imgTstamp":"2008-02-23T09:36:42Z",
"pageURL":"http://www.worksfromthecave.blogspot.com/",
"collection":["AWP1"],
"caption":["great"],

To index:
"id":"a4236137f5455edd8436a5d122c366fa62ba91139f45895f447565b3a6b926bb",
"imgSrc":"http://bp4.blogger.com/_v4YQruRZWLI/RxTDBM9FL5I/AAAAAAAALY/YJbzjdOKH0/s320/2.jpg",
"pageTstamp":"2004-02-15T08:40:21Z",
"imgTstamp":"2009-02-23T09:36:42Z",
"pageURL":"http://www.worksfromthecave.sapo.pt/",
"collection":["AWP3"],
"caption":["fantastic"]
Images in multiple pages

Indexed:
"id":"a4236137f5455edd8436a5d122c366fa62ba91139f45895f447565b3a6b926bb",
"imgSrc":"http://bp3.blogger.com/_v4YQruRZWLI/RxTDBM9FL5I/AAAAAAAALY/YJbzjdOKH0/s320/2.jpg",
"pageTstamp":"2008-02-15T08:40:21Z",
"imgTstamp":"2008-02-23T09:36:42Z",
"pageURL":"http://www.worksfromthecave.blogspot.com/",
"collection":["AWP1"],
"caption":["great"],

To index:
"id":"a4236137f5455edd8436a5d122c366fa62ba91139f45895f447565b3a6b926bb",
"imgSrc":"http://bp4.blogger.com/_v4YQruRZWLI/RxTDBM9FL5I/AAAAAAAALY/YJbzjdOKH0/s320/2.jpg",
"pageTstamp":"2004-02-15T08:40:21Z",
"imgTstamp":"2009-02-23T09:36:42Z",
"pageURL":"http://www.worksfromthecave.sapo.pt/",
"collection":["AWP3"],
"caption":["fantastic"]
Images in multiple pages

Final:
"id": "a4236137f5455edd8436a5d122c366fa62ba91139f45895f447565b3a6b926bb",
"imgSrc": "http://bp4.blogger.com/_v4YQuRZWLI/RxTDBM9FL5I/AAAAAAAAALY/YJbTbdOKH0/s320/2.jpg",
"pageTstamp": "2004-02-15T08:40:21Z",
"imgTstamp": "2009-02-23T09:36:42Z",
"pageURL": "http://www.worksfromthecave.sapo.pt/",
"collection": ["AWP1", "AWP3"],
"caption": ["great", "fantastic"]
Deduplication selected solution

- After careful examination, we arrived at the 3 deduplication scenarios:
  
a. every page-image pair is a document
b. the oldest page that references the image is the canonical document
c. **oldest page information and image specific information from all pages**
  - keep reference to oldest page
  - Add all new image specific information (title, alt, caption) to the document
  - replace oldest page reference if a new oldest document shows up
Group by digest

Group by Digest
● For each image record in the JSONL, send it to the matching Digest list

Merge metadata
● For each entry in the Digest list, merge metadata for that record to produce an unique record for each image
  ○ Similar to the previous merge by SURT step
● If there are multiple Digests for the same URL:
  ○ Pages are updated to represent the data of the image that is closest in capture time
  ○ Additional image information is added to imgAlt, Title and Caption fields
Map Reduce: Group by digest

JSONL set
IE img HTML HTML
IE img img HTML
IE img

Map
SURT 1
IE img entry
Map
SURT 2
IE img entry
(...)
Map
SURT s
IE img entry

Group by img digest
Hadoop
Merge Metadata

Reduce
DIGEST 1
IE img entry
SURT 1
IE img entry
SURT 2
Reduce
DIGEST 2
IE img entry
SURT 3
(...)
Reduce
DIGEST d
IE img entry
SURT s

Unique img entry
Unique img entry
Unique img entry
Duplicates across collections

- Hadoop processing is performed across per collection
  - To better manage computing resources (e.g. HDFS disk space)
  - Thus, deduplication is only performed on a per-collection basis

- We added an extra “group by digest” step when sending docs to Solr
Summary

1. Find all images in records and find image references in pages

2. Group by SURT
   a. store only pages that have new metadata

3. Regroup by Digest
   a. create new records for images with multiple digests

4. Find best image entry for each image reference

5. Send to Solr
Popularity fields

- Extracting multiple versions of each image and pages opens up a world of possibilities!
  - Find how individual pages and images evolve over time (change digests)!
  - Images that appear in more than one page are more or less relevant?
  - Images that change metadata often are less relevant?
  - ....
Metadata: Popularity fields

**matchingImages**
- number of times the image was crawled (by image content digest)

**matchingPages**
- number of times the image was referenced on `<img>` tags, css or JS

**imagesInOriginalPage**
- number of images in the oldest page

**imageMetadataChanges**
- number of times that the image metadata (alt, title or caption) changes

**pageMetadataChanges**
- number of times that the page metadata (title) changes
Takeways

- **Faster WARC parsing**
  - Fixed *two times pass* and WARC download errors
  - *(3 ms -> less than 0.5 ms per image)*

- **A lot more images found!**
  - We will see how many in the following slides...

- **Multiple pages per image** (current ratio: ~2 per image)

- **Removed unneeded bottlenecks** (MongoDB)

- **Logging the indexing process**
  - Hadoop counters for errors
  - Metadata counters for images found and collected
My predictions in May 2020
Arquivo.pt Image Search (as of Jan 2020)

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexed images</td>
<td>22 million</td>
</tr>
<tr>
<td>Collection count</td>
<td>90</td>
</tr>
<tr>
<td>(W)ARCs</td>
<td>3 million</td>
</tr>
<tr>
<td>(W)ARC sizes</td>
<td>334 TB</td>
</tr>
<tr>
<td>Total collected files</td>
<td>6,000 million</td>
</tr>
<tr>
<td>Total collected images</td>
<td>1,602 million</td>
</tr>
<tr>
<td>Oldest image date</td>
<td>15/04/1994</td>
</tr>
<tr>
<td>Newest image date</td>
<td>14/11/2019</td>
</tr>
<tr>
<td>Daily page views</td>
<td>~87</td>
</tr>
<tr>
<td>Collection</td>
<td>Old Parser</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
</tr>
<tr>
<td>AWP24</td>
<td>865,589</td>
</tr>
<tr>
<td>AWP15</td>
<td>552,275</td>
</tr>
<tr>
<td>FAWP26</td>
<td>213,527</td>
</tr>
<tr>
<td>Tomba</td>
<td>169,308</td>
</tr>
<tr>
<td>BlogsSapo2018</td>
<td>71,668</td>
</tr>
<tr>
<td>Weblog</td>
<td>6,336</td>
</tr>
<tr>
<td>DinisAlves2018</td>
<td>1,215</td>
</tr>
<tr>
<td>DEM-IST</td>
<td>191</td>
</tr>
<tr>
<td>BlocoEsquerda</td>
<td>15</td>
</tr>
<tr>
<td>Takeways</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>~200-650 million images</td>
<td></td>
</tr>
<tr>
<td>~9-28x more images</td>
<td></td>
</tr>
<tr>
<td>~400-1,300 million pages (2/image)</td>
<td></td>
</tr>
<tr>
<td>~18-56x more pages</td>
<td></td>
</tr>
</tbody>
</table>

| Indexed images | 1,880,124 | 23,589,395 | 548,823,437 | 23.27x |
| Crawl/Collection count | 9 | 88 | 427,703,203 | 18.13x |
| (W)ARCS | 1,880,124 | 43,742,373 | 548,823,437 | 23.27x |
| (W)ARC sizes | 21.43 TB | 336.47 TB | 686,806,771 | 29.12x |
| Total collected files | 408,230,995 | 6,086,768,283 | 652,203,512 | 27.65x |
Takeways

<table>
<thead>
<tr>
<th>Category</th>
<th>Indexed images</th>
<th>Crawl/Collection count</th>
<th>(W)ARCS</th>
<th>(W)ARC sizes</th>
<th>Total collected files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1,880,124</td>
<td>9</td>
<td>23,589,395</td>
<td>21.43 TB</td>
<td>400,230,995</td>
</tr>
<tr>
<td></td>
<td>23,589,395</td>
<td>88</td>
<td>548,823,437</td>
<td>336.47 TB</td>
<td>6,086,768,283</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,086,768,283</td>
<td>661,310,669</td>
<td>652,203,512</td>
</tr>
<tr>
<td>~200-650 million images</td>
<td>43,742,373</td>
<td></td>
<td>548,823,437</td>
<td>686,806,771</td>
<td>652,203,512</td>
</tr>
<tr>
<td>~9-28x more images</td>
<td></td>
<td></td>
<td>23,589,395</td>
<td>686,806,771</td>
<td>652,203,512</td>
</tr>
<tr>
<td>~400-1,300 million pages (2/image)</td>
<td></td>
<td></td>
<td>23,589,395</td>
<td>686,806,771</td>
<td>652,203,512</td>
</tr>
<tr>
<td>~18-56x more pages</td>
<td></td>
<td></td>
<td>23,589,395</td>
<td>686,806,771</td>
<td>652,203,512</td>
</tr>
</tbody>
</table>
Takeways

**654 million images**

1,880,124 -> 43,742,373

29x more images

**1,252 million pages (1.91/image)**

Indexed images 1,880,124 23,589,395 548,823,437 23.27x
Crawl/Collection count 9 88 427,703,203 18.13x
(W)ARCS 21 43 TB 336.47 TB 686,806,771 29.12x
(W)ARC sizes 654 million images 55x more pages
Total collected files 408,230,995 8,086,789,283 652,203,512 27.65x
But Arquivo.pt kept growing on 2020
### Takeways

**+ 317 million images in one year (2019)**

- Indexed images: 1,880,124 -> 43,742,373
- Crawl/Collection count: 9 -> 88
- (W)ARCS: 3,414,742 -> ???
- (W)ARC sizes: 336.47 TB -> ???

**48% growth**

**+ 610 million pages in one year (2019)**

- Total collected files: 6,086,768,283 -> 652,203,512

**49% growth**

---

*Note: The values marked with `??` are not provided in the document.*
Takeways

971 million images
1,880,124 -> 43,742,373
42x more images

1,862 million pages (1.91/image)

Indexed images
1,880,124

Crawl/Collection count
9

(W)ARCS
23,589,395

(W)ARC sizes
336.47 TB

Total collected files
403,256,993

971 million images
42x more images

81x more pages
## Impact of deduplication

<table>
<thead>
<tr>
<th>Number of documents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a</strong> 1,862 million image-page pair documents</td>
</tr>
<tr>
<td><strong>b</strong> 584 million unique documents (971 million before deduplication across collections)</td>
</tr>
<tr>
<td><strong>c</strong> <em>584 million</em> documents, containing information from all 1,862 million image-page pairs</td>
</tr>
</tbody>
</table>

How will we index these *584 million* documents?
Current Solr indexing architecture

Current image index has **31 million** documents
(22,881,688 plus some special crawls we added in 2020)

on one 20 core, 40 thread server with 512 GB RAM*
* one server per branch, two redundant branches

running Solr 6.3 with a 735 GB index
What to do with new data?

Our indexing process resulted in

584 million documents
(expected index size: ~720GB)

Where will we fit all this data?
Arquivo.pt response time guidelines

The 355 rule

- 3 responses per second
- With an average query time below 5 seconds
- For 5 concurrent users

- We are currently performing these experiments
Planning SolrCloud resource allocation

● Expected index size: ~720GB

● SolrCloud servers:
  ○ 8 servers, 4 per branch
    ■ 512GB: p87, p91 (20/40 cores/threads)
    ■ 256GB: p82, p83 (12/24 c/t), p93, p94, p98, p99 (20/40 c/t)
  ○ 2560GB total, 1280GB per branch

● No SSD, only HDD, but we have more RAM than indexed data
How we configured SolrCloud? - Try 1

- **solr1**
  - shard1
    - 125 GB
    - 97M documents

- **solr2**
  - shard2
    - 125 GB
    - 97M documents

- **solr3**
  - shard3
    - 125 GB
    - 97M documents

- **solr4**
  - shard4
    - 125 GB
    - 97M documents
Solr performance factors

- Available RAM for index file caching
  - slowdown happens when index size > RAM
Solr performance factors

- Available RAM for index file caching
  - slowdown happens when index size > RAM or
  - disk I/O skyrockets
  - and that is basically it
  - CPU or network are not the current bottleneck
How we configured SolrCloud? - Try 2

Server 1
- shard1_1
  - 19.7 GB
  - 18M documents

- ... (omitted)

- shard1_8
  - 19.7 GB
  - 18M documents

Server 2
- shard2_1
  - 19.7 GB
  - 18M documents

- ... (omitted)

- shard2_8
  - 19.7 GB
  - 18M documents

Server 3
- shard3_1
  - 19.7 GB
  - 18M documents

- ... (omitted)

- shard3_8
  - 19.7 GB
  - 18M documents

Server 4
- shard4_1
  - 19.7 GB
  - 18M documents

- ... (omitted)

- shard4_8
  - 19.7 GB
  - 18M documents
How we configured SolrCloud? - Try 2

Server 1
- Solr 1
  - shard1_1
    - 19.7 GB
    - 18M documents
  - shard1_8
    - 19.7 GB
    - 18M documents

Server 2
- Solr 2
  - shard2_1
    - 19.7 GB
    - 18M documents
  - shard2_8
    - 19.7 GB
    - 18M documents

Server 3
- Solr 3
  - shard3_1
    - 19.7 GB
    - 18M documents
  - shard3_8
    - 19.7 GB
    - 18M documents

Server 4
- Solr 4
  - shard4_1
    - 19.7 GB
    - 18M documents
  - shard4_8
    - 19.7 GB
    - 18M documents
### How we configured SolrCloud? - Try 2

<table>
<thead>
<tr>
<th>Server 1</th>
<th>Server 2</th>
<th>Server 3</th>
<th>Server 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Solr 0</strong></td>
<td><strong>Solr 1</strong></td>
<td><strong>Solr 2</strong></td>
<td><strong>Solr 3</strong></td>
</tr>
<tr>
<td>No shards, used as entrypoint for user queries</td>
<td>Shard 1</td>
<td>Shard 9</td>
<td>Shard 17</td>
</tr>
<tr>
<td>19.7 GB</td>
<td>19.7 GB</td>
<td>19.7 GB</td>
<td>19.7 GB</td>
</tr>
<tr>
<td>18M documents</td>
<td>18M documents</td>
<td>18M documents</td>
<td>18M documents</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Shard 8</td>
<td>Shard 16</td>
<td>Shard 24</td>
<td>Shard 32</td>
</tr>
<tr>
<td>19.7 GB</td>
<td>19.7 GB</td>
<td>19.7 GB</td>
<td>19.7 GB</td>
</tr>
<tr>
<td>18M documents</td>
<td>18M documents</td>
<td>18M documents</td>
<td>18M documents</td>
</tr>
</tbody>
</table>

- **Server 1**
  - **Solr 0**: No shards, used as entrypoint for user queries
  - **Solr 1**: Shard 1, 19.7 GB, 18M documents
  - **Shard 8**: 19.7 GB, 18M documents

- **Server 2**
  - **Solr 1**: Shard 9, 19.7 GB, 18M documents
  - **Shard 16**: 19.7 GB, 18M documents

- **Server 3**
  - **Solr 2**: Shard 17, 19.7 GB, 18M documents
  - **Shard 24**: 19.7 GB, 18M documents

- **Server 4**
  - **Solr 3**: Shard 25, 19.7 GB, 18M documents
  - **Shard 32**: 19.7 GB, 18M documents
How to test?

- Search with increasing concurrent users
  - 1, 3, 5, 10, 20, 50 concurrent users

- For a set period of time
  - 5 minutes
How to select realistic queries?

● Two sets of queries:
  ○ User queries extracted from logs
  ○ Random pairs of Portuguese words

● Warmup the index using 50 queries

● Query for 5 minutes and parse the results
(Fresh off the press) results

Single user, random queries (pairs of portuguese words)

<table>
<thead>
<tr>
<th>Label</th>
<th># Samples</th>
<th>Average</th>
<th>Median</th>
<th>90% Line</th>
<th>95% Line</th>
<th>99% Line</th>
<th>Min</th>
<th>Maximum</th>
<th>Error %</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Req...</td>
<td>1004</td>
<td>322</td>
<td>380</td>
<td>460</td>
<td>500</td>
<td>691</td>
<td>50</td>
<td>3477</td>
<td>0.00%</td>
<td>2.5/sec</td>
</tr>
</tbody>
</table>

50 users, random queries (pairs of portuguese words)

<table>
<thead>
<tr>
<th>Label</th>
<th># Samples</th>
<th>Average</th>
<th>Median</th>
<th>90% Line</th>
<th>95% Line</th>
<th>99% Line</th>
<th>Min</th>
<th>Maximum</th>
<th>Error %</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Req...</td>
<td>5066</td>
<td>2726</td>
<td>2769</td>
<td>4856</td>
<td>5304</td>
<td>6210</td>
<td>25</td>
<td>9090</td>
<td>2.17%</td>
<td>16.8/sec</td>
</tr>
</tbody>
</table>
Tips and parameters

- vmtouch tool to force OS to keep index files in RAM
- Heap size: 31GB
  - Smaller sizes made Solr crash on parallel query situations
  - Larger sizes means Java can’t use compressed pointers
How we configured SolrCloud?

```
solr1_1
  shard1_1
    20 GB
    18M documents

...  

solr1_8
  shard1_8
    20 GB
    18M documents
```
How we configured SolrCloud?

- **solr1_1**
  - shard1_1
  - 20 GB
  - 18M documents

- **solr2_1**
  - shard2_1
  - 20 GB
  - 18M documents

- **solr3_1**
  - shard3_1
  - 20 GB
  - 18M documents

- **solr4_1**
  - shard4_1
  - 20 GB
  - 18M documents

- **solr_0**
  - No shards, will only take requests and aggregate results from other instances
Future problems: Migrate page search to SolrCloud

- Currently, we have an highly customized version of Lucene optimized not to search the full posting lists

- Scale
  - 6-7,000 million documents
  - 5 servers with 4.5TB of RAM in total
Modernize NSFW libs
What about NSFW content?

- Arquivo.pt captures pages and images from all over the web
- This includes content that may be offensive to users
- Arquivo.pt uses an image based NSFW content classifier
- Images marked as NSFW are filtered by default from image search results
Arquivo’s NSFW classifier

- Based in Keras ResNet
  - [https://github.com/GantMan/nsfw_model](https://github.com/GantMan/nsfw_model)
  - Reported 93% precision
  - Measurements in our test collection match it at about 90% precision
  - ~250 images per second per GPU times 2 GPUs

- Extra features
  - Detect whether the image is a drawing/vectorial or picture
NSFW

- **Antigo:** ~40 imagens por segundo
  - estimativa para processamento dos novos dados (assumindo 2 GPU): ~150 dias

- **Novo:** ~250 imagens por segundo
  - estimativa para processamento dos novos dados (assumindo 2 GPU): ~30 dias

- **Antigo:**
  - Precision  Recall  F1
  - 0.92  0.94  0.93

- **Novo:**
  - Precision  Recall  F1
  - 0.94  0.88  0.91
Architecture/Pipeline
Queue

The collection becomes available for search in Arquivo.pt

Queue

WARC collection → Hadoop queue → Find Images → Merge Dups → NSFW queue → Classify NSFW → Validate Manually

Hadoop Cluster

Search Index → Index Images → Send to SolrCloud

SolrCloud Cluster

NSFW Cluster

POST queue
Relevance assessment
Results on TestCollection (2020)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Arquivo.pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP</td>
<td>0.5471</td>
</tr>
<tr>
<td>nDCG@1</td>
<td>0.6800</td>
</tr>
<tr>
<td>nDCG@5</td>
<td>0.5480</td>
</tr>
<tr>
<td>nDCG@10</td>
<td>0.4800</td>
</tr>
<tr>
<td>nDCG@20</td>
<td>0.4270</td>
</tr>
<tr>
<td>P@1</td>
<td>0.6800</td>
</tr>
<tr>
<td>P@5</td>
<td>0.5930</td>
</tr>
<tr>
<td>P@10</td>
<td>0.5703</td>
</tr>
<tr>
<td>P@20</td>
<td>0.5834</td>
</tr>
<tr>
<td>S@1</td>
<td>0.6800</td>
</tr>
<tr>
<td>S@5</td>
<td>0.8200</td>
</tr>
<tr>
<td>S@10</td>
<td>0.8600</td>
</tr>
<tr>
<td>S@20</td>
<td>0.9000</td>
</tr>
</tbody>
</table>
Summary of what changed in 2020?

● More metadata per image
  ○ All pages that mention the image are parsed
  ○ Heuristic extraction of image captions from the HTML page structure
  ○ Additional features extracted from the HTML and images

● Improved NSFW image processing
  ○ 7x faster processing (40 -> 280 images per second)
  ○ Returns more image information for ranking (e.g. drawing vs. photo)

● Improved indexing architecture and processing
  ○ Removed MongoDB dependency
  ○ Ensure all archived images and pages are parsed
  ○ Find images in <a> links, CSS and JS code

● Distributed search index
  ○ Transition from single node Solr to distributed SolrCloud architecture
  ○ Improved schema so that the index only grows by 32% when covering 81x more images
Plan for the future

- **Deal with images that have no metadata**
  - Cannot find pages for 300+ million images
  - Deep Image classification, **tag extraction**

- **Content based hashes**
  - Similar images show up all over the place (different resolutions and formats)
  - Find and deduplicate **near duplicates**

- **Improve Solr ranking**
  - Use the newly extracted popularity features
## 2020 vs 2021

<table>
<thead>
<tr>
<th>January 2020</th>
<th>January 2021</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>22 million images in pages (only one version of the image is indexed per collection)</td>
<td>1,862 million images in pages</td>
<td><strong>81x</strong> more images in pages parsed</td>
</tr>
<tr>
<td>967 million image files</td>
<td><strong>42x</strong> more image files parsed</td>
<td></td>
</tr>
<tr>
<td>17 million deduplicated documents</td>
<td>584 million deduplicated documents</td>
<td><strong>33x</strong> more unique images, removing duplicates across collections</td>
</tr>
<tr>
<td>49% have image metadata (imgAlt, imgTitle)</td>
<td>99%+ have image metadata (imgAlt, imgTitle, imgCaption)</td>
<td><strong>+51 p.p.</strong> images can be found with relevant contextual information</td>
</tr>
<tr>
<td>~570 GB search index</td>
<td>~750 GB search index</td>
<td><strong>Only 32% larger</strong> after a <strong>813% increase</strong> in information parsed</td>
</tr>
<tr>
<td>1 Solr server</td>
<td>4 SolrCloud servers</td>
<td><strong>Only 3 more nodes</strong> after a <strong>813% increase</strong> in information parsed</td>
</tr>
</tbody>
</table>
## 2020 vs 2021

<table>
<thead>
<tr>
<th>January 2020</th>
<th>January 2021</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>22,881,688 images in pages (only one version of the image is indexed per collection)</td>
<td>1,862,311,456 images in pages</td>
<td><strong>81.39x</strong> more images in pages parsed</td>
</tr>
<tr>
<td>967,184,126 image files</td>
<td>1,862,311,456 images in pages</td>
<td><strong>42.26x</strong> more image files parsed</td>
</tr>
<tr>
<td>17,643,047 deduplicated documents</td>
<td>584,242,176 deduplicated documents</td>
<td><strong>33.11x</strong> more unique images, removing duplicates across collections</td>
</tr>
<tr>
<td>48.7% have image metadata (imgAlt, imgTitle)</td>
<td>99.6% have image metadata (imgAlt, imgTitle, imgCaption)</td>
<td>+50.9 p.p. images can be found with relevant contextual information</td>
</tr>
<tr>
<td>~570 GB search index</td>
<td>~750 GB search index</td>
<td><strong>Only 32% larger</strong> after a <strong>813% increase</strong> in information parsed</td>
</tr>
<tr>
<td>1 Solr server</td>
<td>4 SolrCloud servers</td>
<td><strong>Only 3 more nodes</strong> after a <strong>813% increase</strong> in information parsed</td>
</tr>
</tbody>
</table>
Ranking features for 2021

**imgCaption**
- portion of the HTML page text that is closest to the image

**matchingImages**
- number of times the image was crawled (by image content digest)

**matchingPages**
- number of times the image was referenced on `<img>` tags, css or JS

**imagesInOriginalPage**
- number of images in the oldest page

**imageMetadataChanges**
- number of times that the image metadata (alt, title or caption) changes

**pageMetadataChanges**
- number of times that the page metadata (title) changes

**drawing/photo**
- whether the image is a drawing or a photo