Towards

Evaluating the evolution of Wikipedia’s navigability

Project in communication systems II

June 12, 2017

Emanuele Bugliarello
emanuele.bugliarello@epfl.ch
(SC-MA4)

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

dlab

Motivation

- **People are regularly faced with navigating information spaces**
  Information is usually spread over different interconnected sources

- **This type of navigation can be mapped to a search in a graph**
  Nodes represent pieces of knowledge; edges indicate connections

- **Lack of a global view of the underlying network**
  We only get access to local information

- **Decentralized search in giant networks**
  People could easily get lost and we cannot rely on well-known results

Goals

- **Study which properties of a network have the largest impact on navigability**
  More links: more shortcuts; higher risk of users getting lost

- **Infer a model to help people in exploratory searches**
  Information usually gathered from sources not known in advance

- **Using Wikipedia as our validation domain**
  - Rich knowledge database
  - Data of human navigation from *Wikispeedia*
  - Entire history of revisions of Wikipedia available

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

Related Work

- **Data management systems to store historical graph data**
  - B. Salzberg and V. Tsotras.
    Comparison of access methods for time-evolving data.
    ACM Computing Surveys, 1999.

- **Graph properties evolution over time**
  - J. Leskovec, J. Kleinberg, and C. Faloutsos.
    Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations.
    ACM SIGKDD, 2005.

- **Decentralized search in networks**
  - R. West and J. Leskovec.
    Human Wayfinding in Information Networks.
    WWW, 2012.
  - R. West and J. Leskovec.
    Automatic Versus Human Navigation in Information Networks.
    ICWSM, 2012.

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

System Overview

- **dlab-server (or simply server)**
  `iccluster111.iccluster.epfl.ch`
  Single Linux machine: 48 cores, 256 GB RAM
  Data folder: `/scratch/bugliare/`

- **cluster**
  `hadoop.iccluster.epfl.ch`
  7 nodes: 266 VCores, 1.63 TB RAM in total
  Data folder: `hdfs:///user/bugliare/data/`
  Pay-per-use

- *`iccluster050.iccluster.epfl.ch`*
  Machine to move data from cluster's `HDFS` to server's `/dlabdata1/`

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

Overview

- Datasets are Spark DataFrames
  - Stored as Parquet files
  - Compressed with Snappy

- Two processing stages:
  - General Transformations
  - Project-related Transformations

Original Data

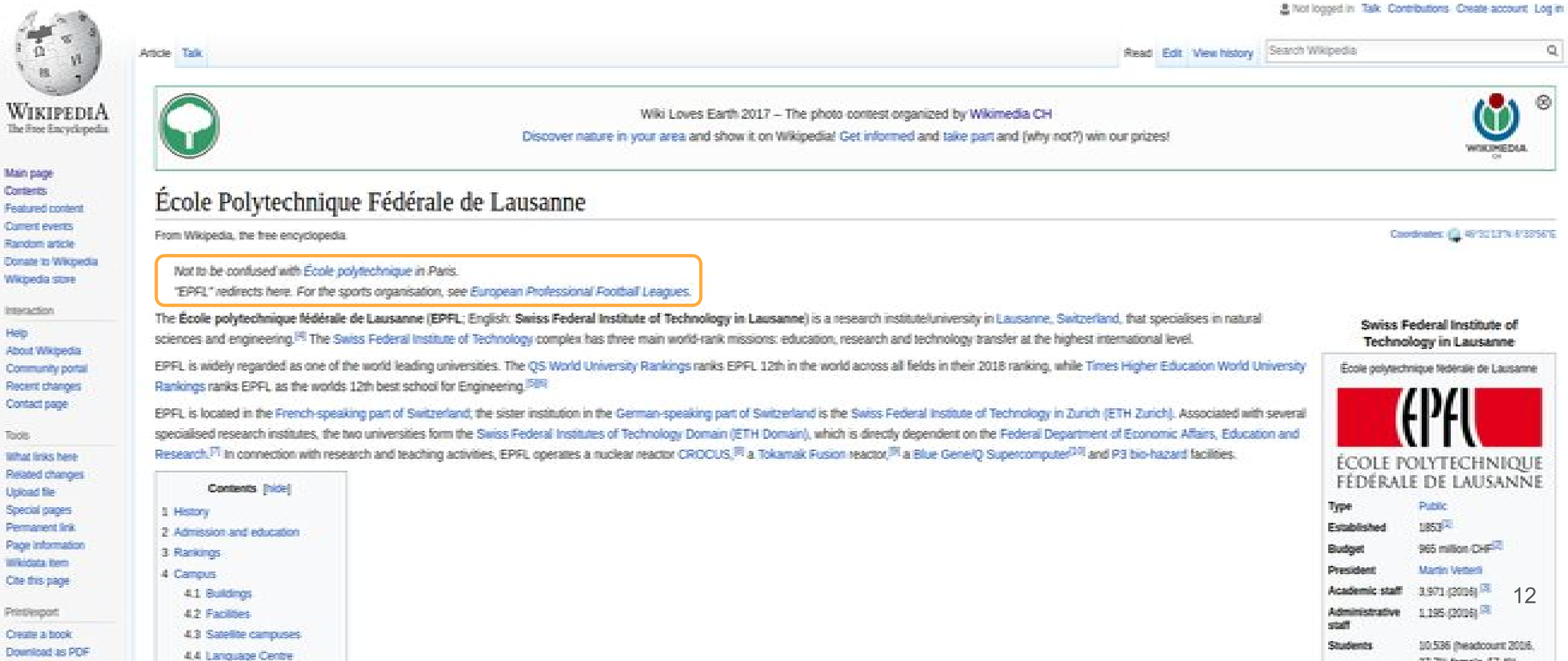
- Available at `hdfs:///datasets/wikipedia/en-oct-2016`

⚠ Cannot work with the entire dataset at once

redirect: *UTF-8*	null
ns: *UTF-8*	4
title: *UTF-8*	Wikipedia:Bot req...
id: *UTF-8*	912023
sha1: *UTF-8*	4qpzrtsom7ls28291...
revision_id: *UTF-8*	336934282
parentid: *UTF-8*	336930575
model: *UTF-8*	wikitext
text: *UTF-8*	{{Wikipedia progr...
text_xmlspace: *UTF-8*	preserve
ip: *UTF-8*	null
timestamp: *int96*	2010-01-10 05:06:...

Extracting hyperlinks from text

- Standard hyperlinks
- Hatnotes



Wiki Loves Earth 2017 – The photo contest organized by Wikimedia CH
Discover nature in your area and show it on Wikipedia! Get informed and take part and (why not?) win our prizes!

École Polytechnique Fédérale de Lausanne

From Wikipedia, the free encyclopedia

Coordinates: 46°31′13″N 6°33′56″E

Not to be confused with École polytechnique in Paris.
"EPFL" redirects here. For the sports organisation, see European Professional Football Leagues.

The **École polytechnique fédérale de Lausanne** (**EPFL**; English: **Swiss Federal Institute of Technology in Lausanne**) is a research institute/university in Lausanne, Switzerland, that specialises in natural sciences and engineering.[4] The Swiss Federal Institute of Technology complex has three main world-rank missions: education, research and technology transfer at the highest international level.

EPFL is widely regarded as one of the world leading universities. The QS World University Rankings ranks EPFL 12th in the world across all fields in their 2018 ranking, while Times Higher Education World University Rankings ranks EPFL as the worlds 12th best school for Engineering.[5][6]

EPFL is located in the French-speaking part of Switzerland; the sister institution in the German-speaking part of Switzerland is the Swiss Federal Institute of Technology in Zurich (ETH Zurich). Associated with several specialised research institutes, the two universities form the Swiss Federal Institutes of Technology Domain (ETH Domain), which is directly dependent on the Federal Department of Economic Affairs, Education and Research.[7] In connection with research and teaching activities, EPFL operates a nuclear reactor CROCUS,[8] a Tokamak Fusion reactor,[9] a Blue Gene/Q Supercomputer[10] and P3 bio-hazard facilities.

Contents [hide]

Swiss Federal Institute of Technology in Lausanne

École polytechnique fédérale de Lausanne

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Type	Public
Established	1853[1]
Budget	965 million CHF[2]
President	Martin Vetterli
Academic staff	3,971 (2016)[3]
Administrative staff	1,195 (2016)[3]
Students	10,536 (headcount 2016,

Extracting hyperlinks from text: standard links

- `[[Texas]]` → https://en.wikipedia.org/wiki/Texas

  ... claimed the territory of Texas in the 18th century as ...

- `[[Texas|Lone Star State]]`→ https://en.wikipedia.org/wiki/Texas

  ... claimed the territory of Lone Star State in the 18th century as ...

Extracting hyperlinks from text: hatnotes (1)

- *“Main article: . . . ”*
- *“For more details on . . . , see . . . ”*
- *“See also . . .”*
- *“Further information: . . . ”*
- *“This page is about . . . For other uses . . . ”*
- *“This page is about . . . It is not to be confused with . . . ”*
- *“For . . . , see . . . ”*
- *“For other uses, see . . . ”*
- *“. . . redirects here. For other uses, see . . . ”*
- *“Not to be confused with . . . ”*
- *“. . . Not to be confused with . . . ”*

Extracting hyperlinks from text: hatnotes (2)

Example (*About* hatnote)

"This page is about ... For other uses ..." inside Wikipedia page PAGETITLE

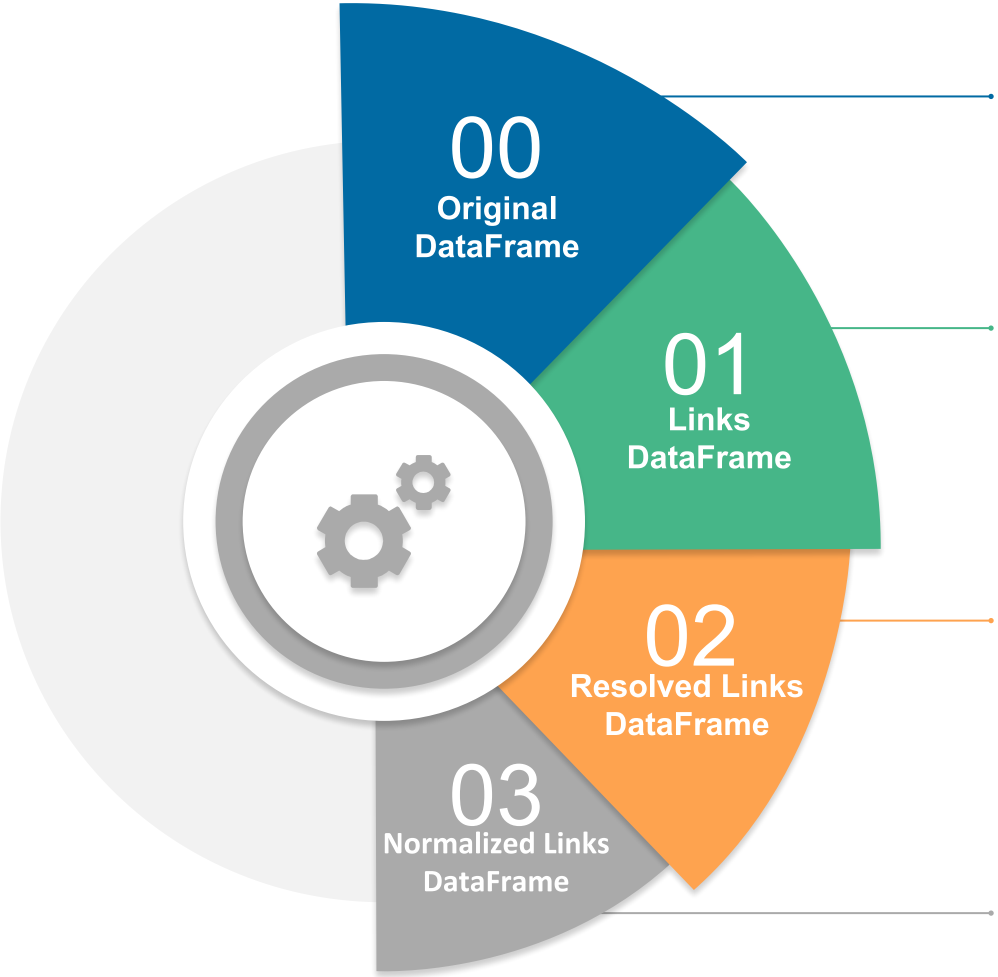
- `{{About|USE1}}`

  This page is about USE1. For other uses, see PAGETITLE (disambiguation)

- `{{About|USE1|USE2|PAGE2{{!}}PAGE2TITLE|and|PAGE3#SUBSECTION|other uses}}`

  This page is about USE1. For USE2, see PAGE2TITLE and PAGE3. For other uses, see PAGETITLE (disambiguation)

General Transformations



DataFrame of Wikipedia revisions. For each revision, the whole text (wiki-markup) is stored.

DataFrame of links. It contains two columns, one for standard hyperlinks and one for hatnotes. A column storing the number characters is also added.

DataFrame with normalized titles and resolved redirects pages, substituting each of their occurrences with the corresponding target pages.

DataFrame with columns containing integer values casted to integer and links split into “reachable” and “unreachable” columns.

Links DataFrame

- Keep a counter of each link frequency
- Separate standard hyperlinks to hatnotes
- Extract titles in the redirect column: `{@title=Tautology}` → `Tautology`
- Add `length` column: number of characters in the text of each revision

id: *UTF-8*	912023
title: *UTF-8*	Wikipedia:Bot requests
timestamp: *int96*	2007-01-26 16:54:18
standard_outlinks: *array(array(UTF-8))*	[[Skynet, 1], ...]
hatnotes_outlinks: *array(array(UTF-8))*	[]
length: *int*	29155
redirect: *UTF-8*	null
revision_id: *UTF-8*	103395195
ip: *UTF-8*	null
parentid: *UTF-8*	103389117
ns: *UTF-8*	4

Resolved Links DataFrame

A redirect is a page which automatically sends visitors to another page

Example:

https://en.wikipedia.org/wiki/UK → https://en.wikipedia.org/wiki/United_Kingdom

- Redirect information readily available in the `redirect` column

- A page might change target over time → intensive replacing task:
  For each revision in the DataFrame, determine its view of Wikipedia in terms of redirects

- Normalize titles:
  - `/subpageTitle` → `hostPageTitle/subpageTitle`
  - Replace white spaces by underscores (_)
  - Capitalize first letter

Normalized Links DataFrame (1)

- Cast `id`, `revision_id`, `parentid` and `ns` columns to integer

- Split links in the `standard` and `hatnotes` lists into two lists each: “reachable” and “unreachable” using the timestamps of first revisions
  - Discard links pointing to non-existing pages (e.g., due to typos)
  - Discard links to pages not existing at the moment the revision was created (red links)

Normalized Links DataFrame (2)

id: *int*	24657
title: *UTF-8*	Standard_Chinese
timestamp: *int96*	2016-09-17 02:58:55
standard_outlinks: *array(array(UTF-8))*	[[Compound_(linguistics), 1], ...]
hatnotes_outlinks: *array(array(UTF-8))*	[[Standard_Chinese_(disambiguation), 1]]
standard_outlinks_failed: *array(array(UTF-8))*	[[Wikt:mingzi, 1], [Wikt:goodbye, 2], ...]
hatnotes_outlinks_failed: *array(array(UTF-8))*	[]
length: *int*	49416
redirect: *UTF-8*	null
revision_id: *int*	739788881
ip: *UTF-8*	null
parentid: *int*	739788073
ns: *int*	0

Testing

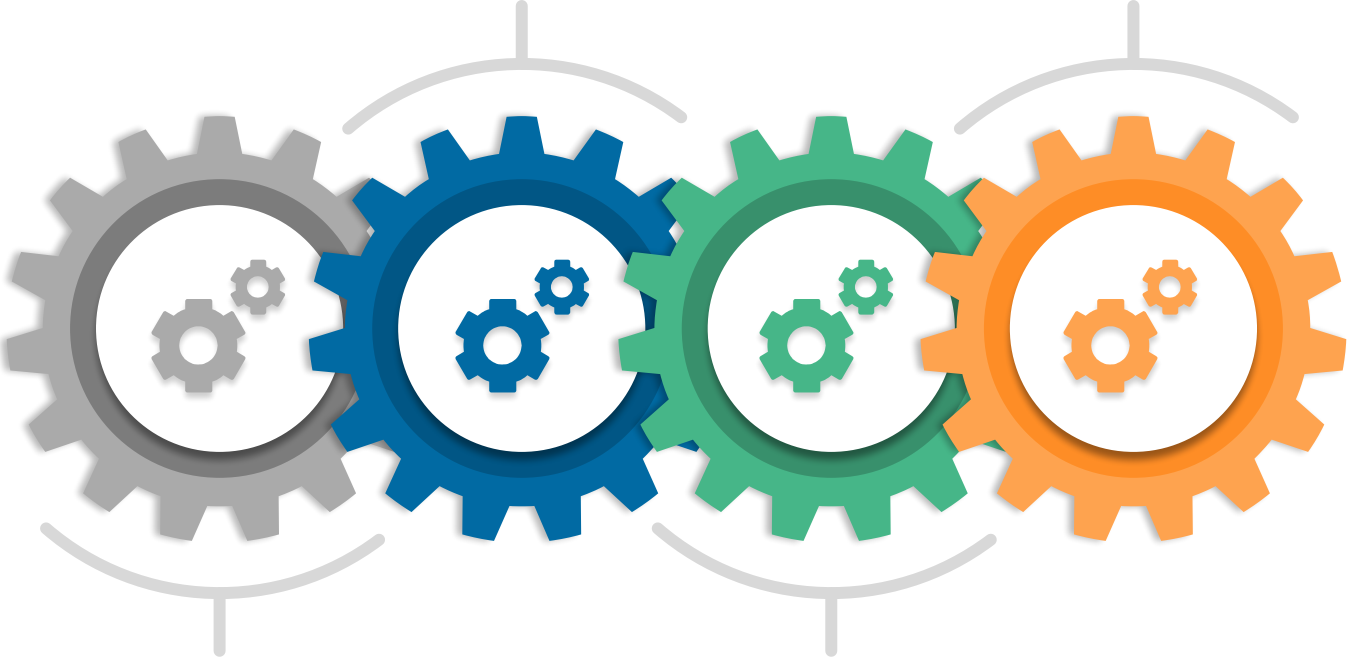
Inspect an article with corner cases:

- Still has red links as of the test date
  → Assert whether they have been placed in any of the `_failed` lists
- Contains links with non-ASCII characters
  → Make sure they have been preserved along the pipeline
- Contains redirects among its links
  → Assert whether they have been correctly resolved

Winning article: “Standard Chinese” (revision 739788881, edited on Sep 17, 2016)
Result: all tests passed ✓

Sanity check: Number of entries in the `Normalized Links DataFrame` is equal to the number of entries in the `Original DataFrame`.

Project-related Transformations



Monthly Links DataFrame

For each article, take the last revision in each month (if any)

Monthly Snapshots Edge Lists

Edge lists of monthly snapshots of the whole network

Normalized Links DataFrame

Monthly Edge Lists

Edge lists of monthly differences

Monthly Links DataFrame

- For each month, keep the revision with the latest timestamp for any of the articles modified in that month
  → At most one entry per month for each article

- Focus only on articles in the main namespace (0)
  → Entries still have links to pages in other namespaces!

Collisions & Other Issues

- Needed mappings: `id → title`, `title → id`
  a. Create a dictionary having ids as keys and titles as values
  b. Invert keys and values and create a dictionary mapping titles to ids

⚠ 123 titles that have each 2 ids associated

- ○ 105 single-letter Unicode character pairs collide due to capitalization
  Example: ⓩ → Ⓩ
  Not a big issue: Most of them redirect to their “normal” representation
- ○ 18 id pairs collide because their title in the `Original DataFrame` is wrong
  Example:
  - Claim: `5702430 → Akalgarh, India`
  - Truth: `5702430 → Akalgarh, Ludhiana`
  ⚠ Incorrect `Original DataFrame`

- Manually map each id to its correct title using Wikipedia’s query API
  → Ensure that the title to id dictionary has the correct mapping

Monthly Edge Lists

- An edge list file contains an edge per line as a source-target ids pair
  - Repeat an edge as many times as its frequency in the original article

- Build independent monthly graphs as edge lists
  - Edge lists only contain edges whose sources have been modified in a given month
  - Only use non-failing links
  - Map titles to ids using dictionary with keys in namespace 0 only
    → Discard links to articles not in the main namespace (`O(1)`)

Monthly Snapshots Edge Lists

Build edge lists representing a snapshot of Wikipedia for a given month

- Start from the first month in the dataset

- Incrementally build dictionary of:

  `article id → latest list of outgoing links`

- Purge sources whose last revision in the `Original DataFrame` is at least one year older than the current month

  → This is used to infer the missing information of removed articles

  → Reasonable results:

  - Number of articles in namespace 0 in the last revision: 4,947,285
  - Real number of pages in Wikipedia: 5,420,384

Resources on the cluster

- Pipeline running time: 8.70 + 23.11 + 20.72 + 11.61 + 34.90 = 99.04 hours
- Pipeline cost: 11.66 + 27.64 + 19.30 + 18.94 + 21.63 = 99.17 CHF

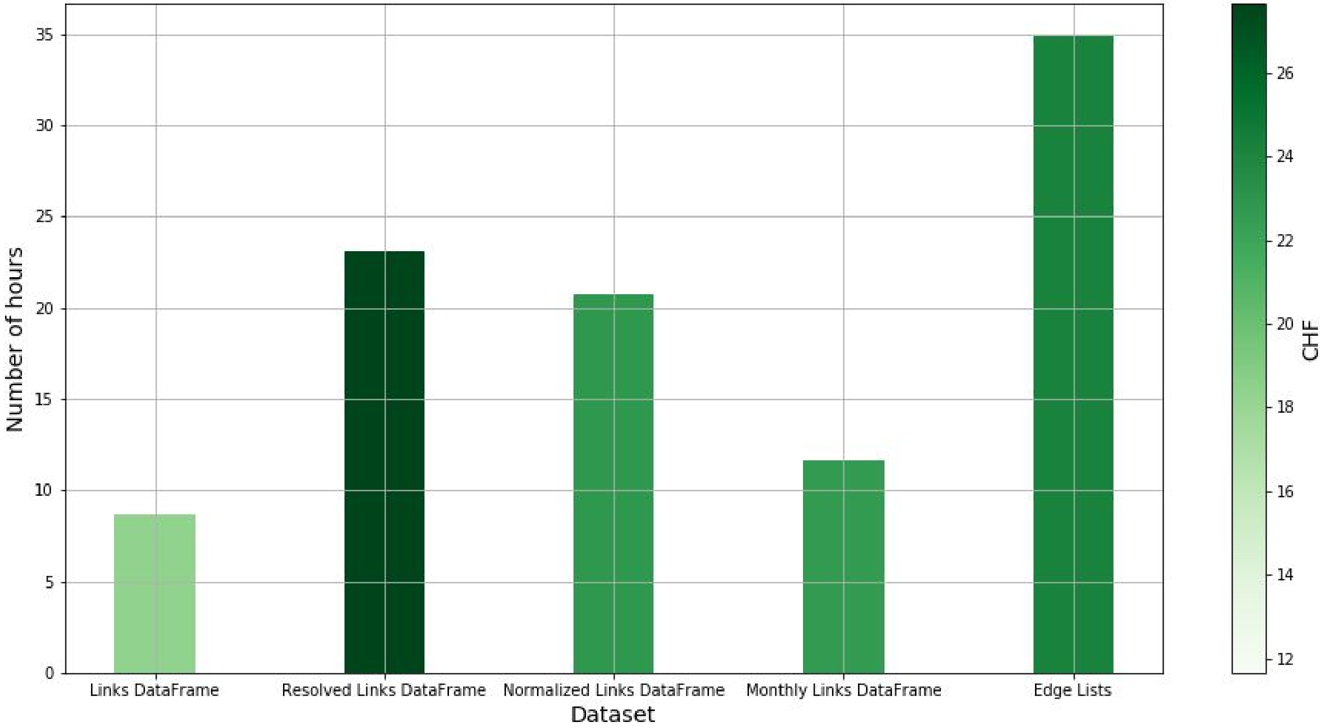


Figure 1: *Time and cost of each processing phase*

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

Last snapshot: preprocessing definition

- Self loops in the network despite removing them during hyperlinks extraction (due to redirects)
  → Remove them

- Many 0 out-degree nodes
  → Remove them

Last snapshot: degree distributions

- Do the degrees follow a power law distribution?

- If yes, how do their parameters evolve over time?

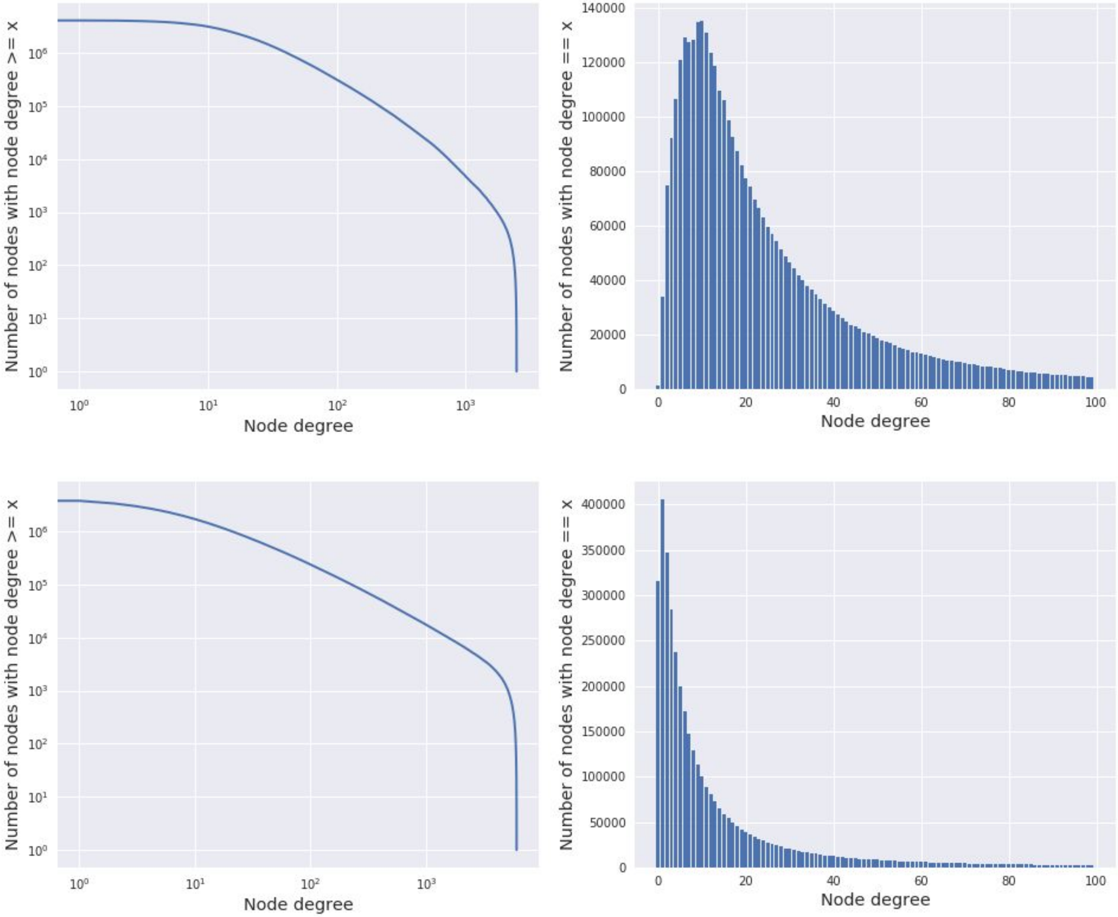


Figure 2: *Outdegree (top) and Indegree (bottom) distributions in the last snapshot of Wikipedia*

Wikipedia over time: nodes evolution

- Before 2012 Wikipedia's growth approximately followed a Gompertz growth model:

$$y(t) = ae^{be^{ct}}$$

  - a = 4378449
  - b = −15.42677
  - c = −0.384124
  - t is the time in years since 1/1/2000 (so 1/1/2010 is t=10.00)

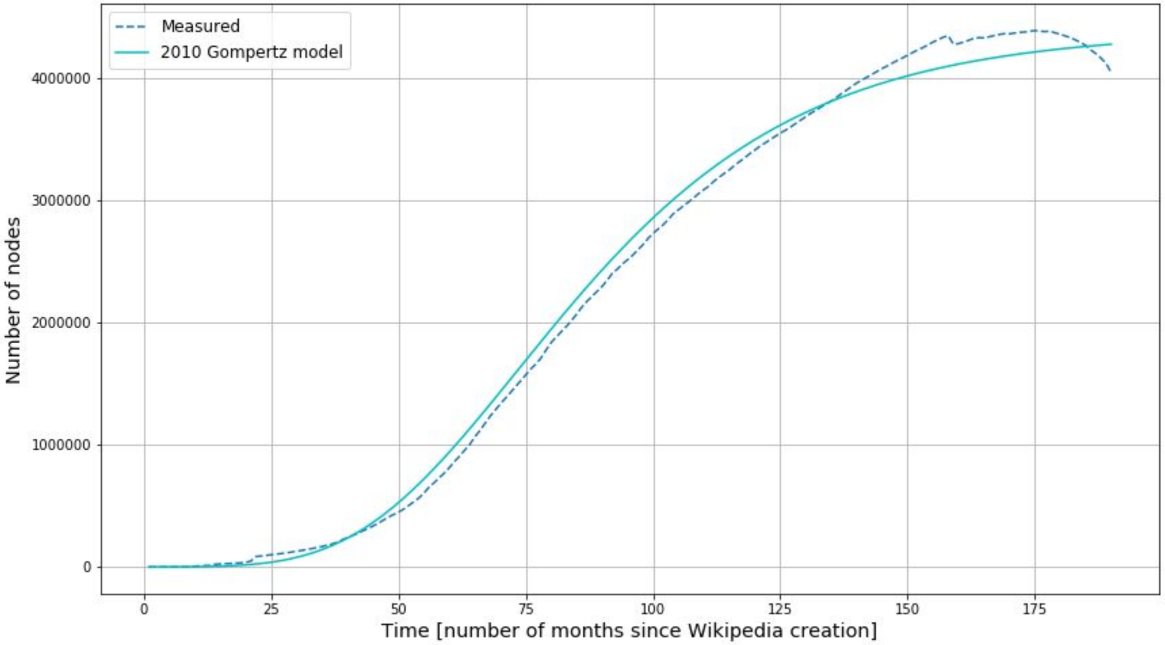


Figure 3: *Evolution of the number of nodes in Wikipedia’s network.*

Wikipedia over time: edges & graph density evolution

- Wikipedia’s graph becomes denser over time

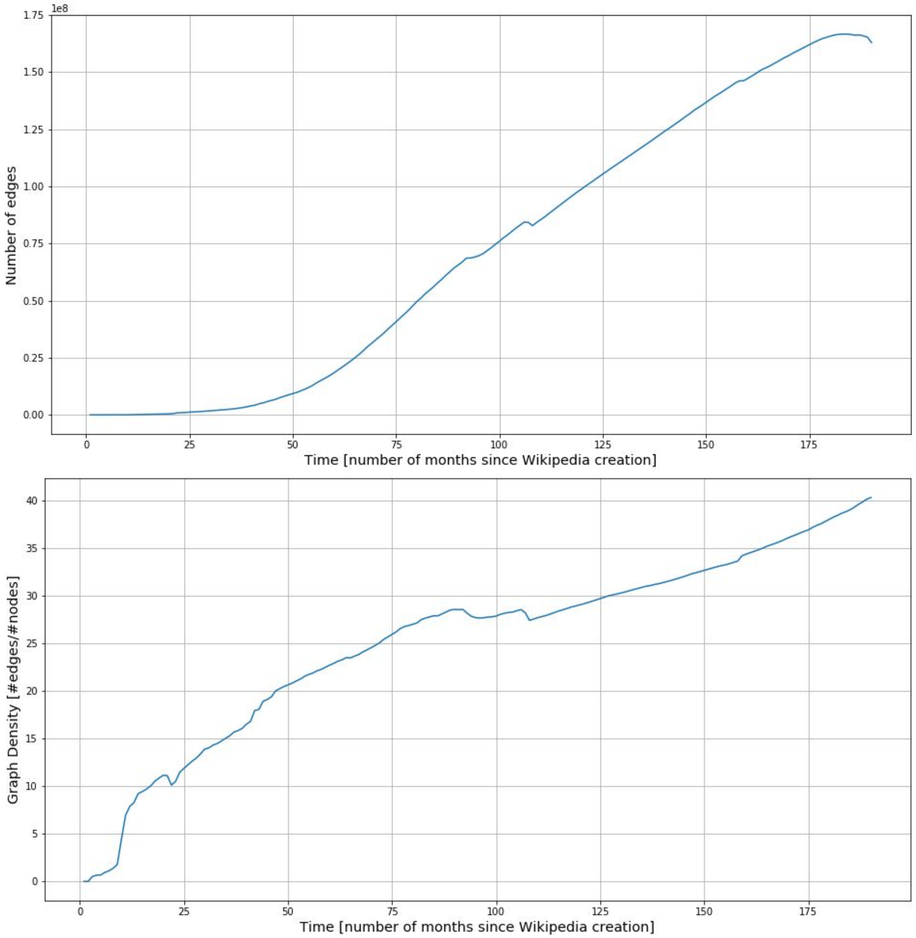


Figure 4: *Evolution of the number of edges (top) and graph density (bottom) in Wikipedia’s network.*

Wikipedia over time: median degrees evolution

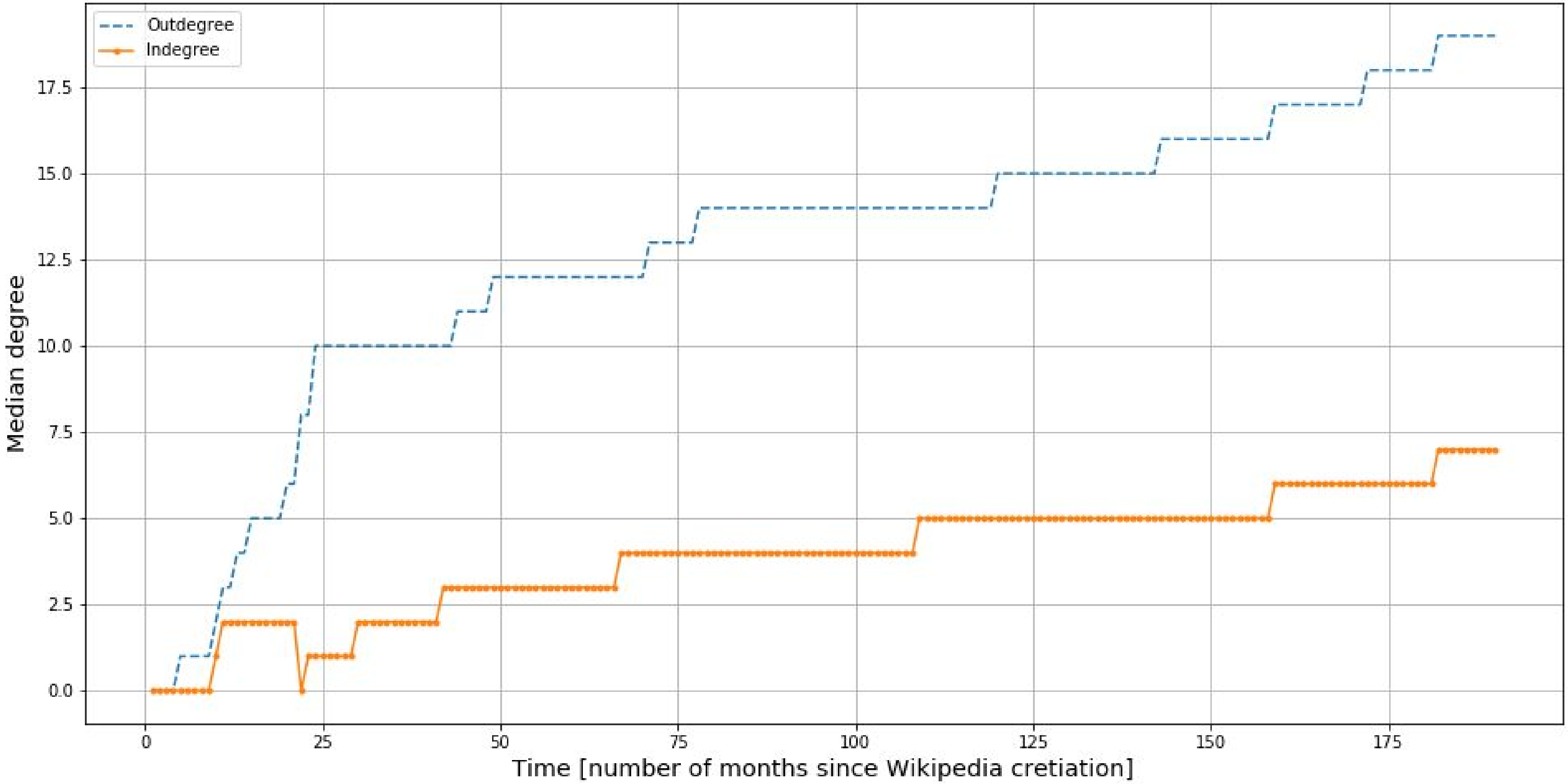


Figure 5: *Evolution of the median degrees in Wikipedia’s network.*

Wikipedia over time: giant connected component evolution

- Giant component size increases over time

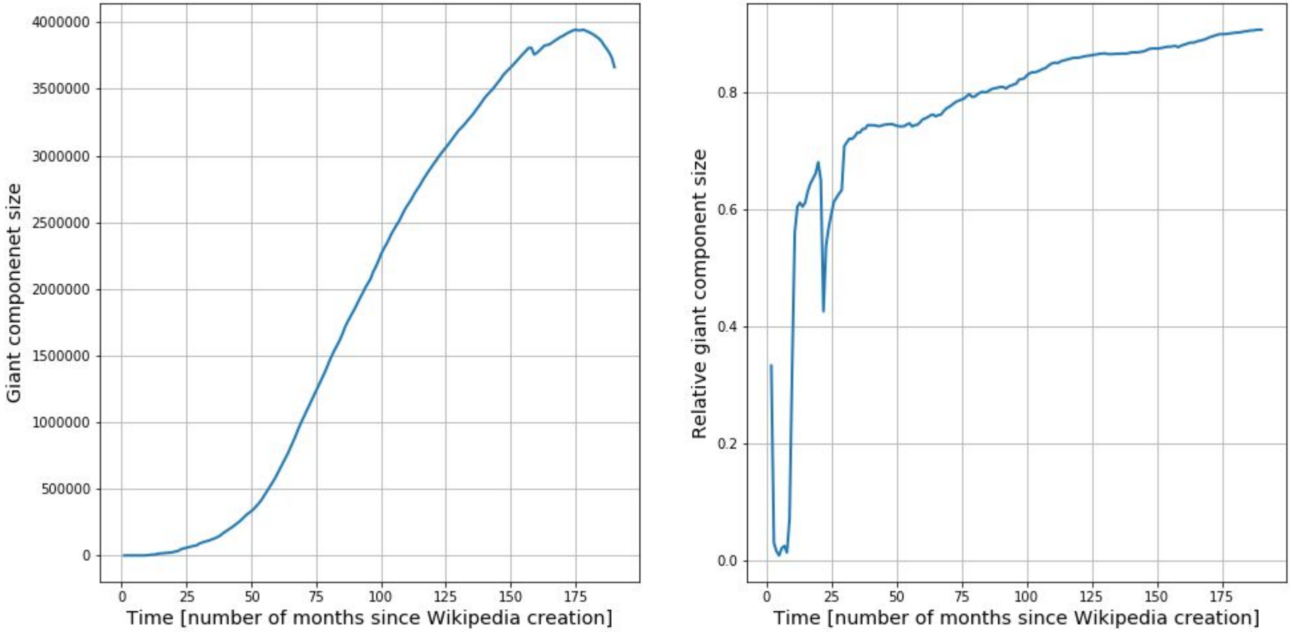


Figure 6: *Evolution of the size of the giant component in absolute values (left) and as a percentage of the total number of nodes in each month (right).*

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

Methodology (1)

- Python scripts to reproduce everything we report on
  IPython notebooks to show results and intermediate steps

- 3 widely commented libraries [> 2500 lines]
  - Wikipedia parsing
  - Data processing
  - Network analysis (interface to `Snap.py`)

- GitHub repository with descriptions for each piece of code

- Report with meticulously described processing phases

Methodology (2)

BIG DATA

BIG TIME

BIG MONEY

- Bash scripts to launch Spark jobs tuned to ask for minimal resources

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

Spark on YARN: overview

- `--num-executors`: number of executors requested
- `--executor-cores`: number of executor cores requested
- `--executor-memory`: executor JVM heap size
- `--conf spark.yarn.executor.memoryOverhead`: determines full memory request to YARN for each executor. Default: max(384, 0.07*spark.executor.memory)
- `--driver-memory` and `--driver-cores`: resources for the application master

- Python is all off-heap memory and does not use the RAM reserved for heap!

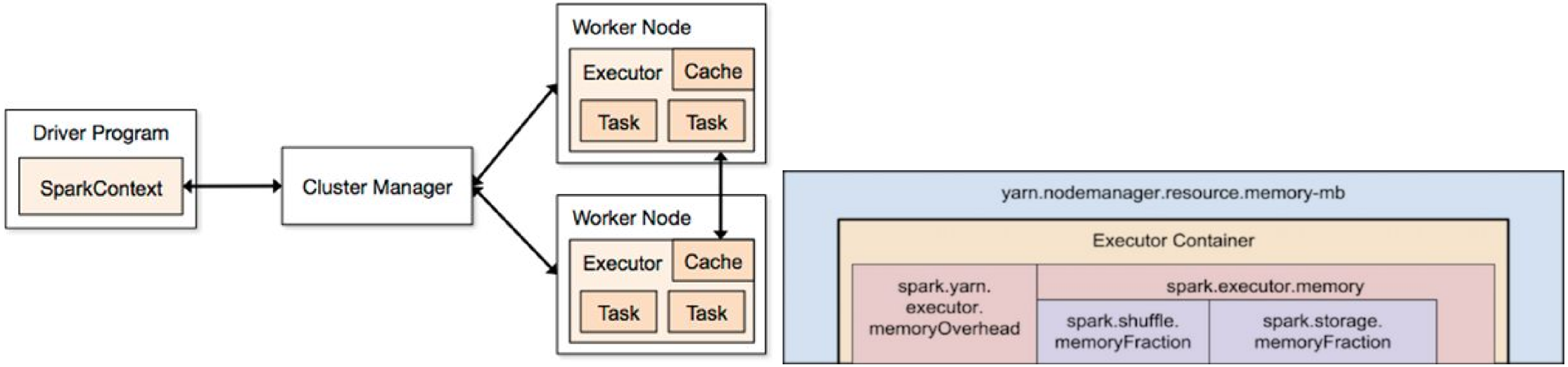


Figure TODO: *Spark architecture and container memory layout*

Spark on YARN: errors & solutions

Memory resources are split among all the cores of each executor

- … Consider boosting spark.yarn.executor.memoryOverhead.
  → `--conf spark.yarn.executor.memoryOverhead=<N_MB>`
- `java.lang.OutOfMemoryError: Java heap space`
  `java.lang.OutOfMemoryError: GC overhead limit exceeded`
  → boost `driver-memory` and/or `executor-memory`
- `java.lang.NullPointerException`
  → Error in the cluster:
  - Service down in one node
  - No storage left in output directory
- Serialized results ... is bigger than spark.driver.maxResultSize
  → `--conf spark.driver.maxResultSize=<N>G`
- If you want to share a large dictionary `dict_`, use: `dict_bc=sc.broadcast(dict_)`
  And access it as: `dict_bc.value`

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

Conclusion

- Generated datasets of resolved & reachable links from Wikipedia revisions despite slow start
  - General transformations applied without any loss of granularity

- Extensively commented code and README files
  → Making our results easily reproducible

- Scripts to avoid spending time and money on tuning Spark

- Early results on Wikipedia’s network evolution over time
  - Graph densifies over time

ToC

- Introduction
- Related Work
- System Overview
- Data, Data & Data
- Results
- Methodology
- On Spark & YARN
- Conclusion
- Future Work

Future Work

Thank you

- **Data processing**
  - Investigate 0 out-degree nodes
  - Add the position of each link in the text
  - Read Spark DataFrame in Parquet files into other platforms (Hadoop)

- **Graph's properties evolution**
  - Diameter
  - Link density per article
  - Indegree saturation time

- **Navigability studies**