PISA
Performant Indexes and Search for Academia

Antonio Mallia
New York University

Michal Siedlaczek
New York University

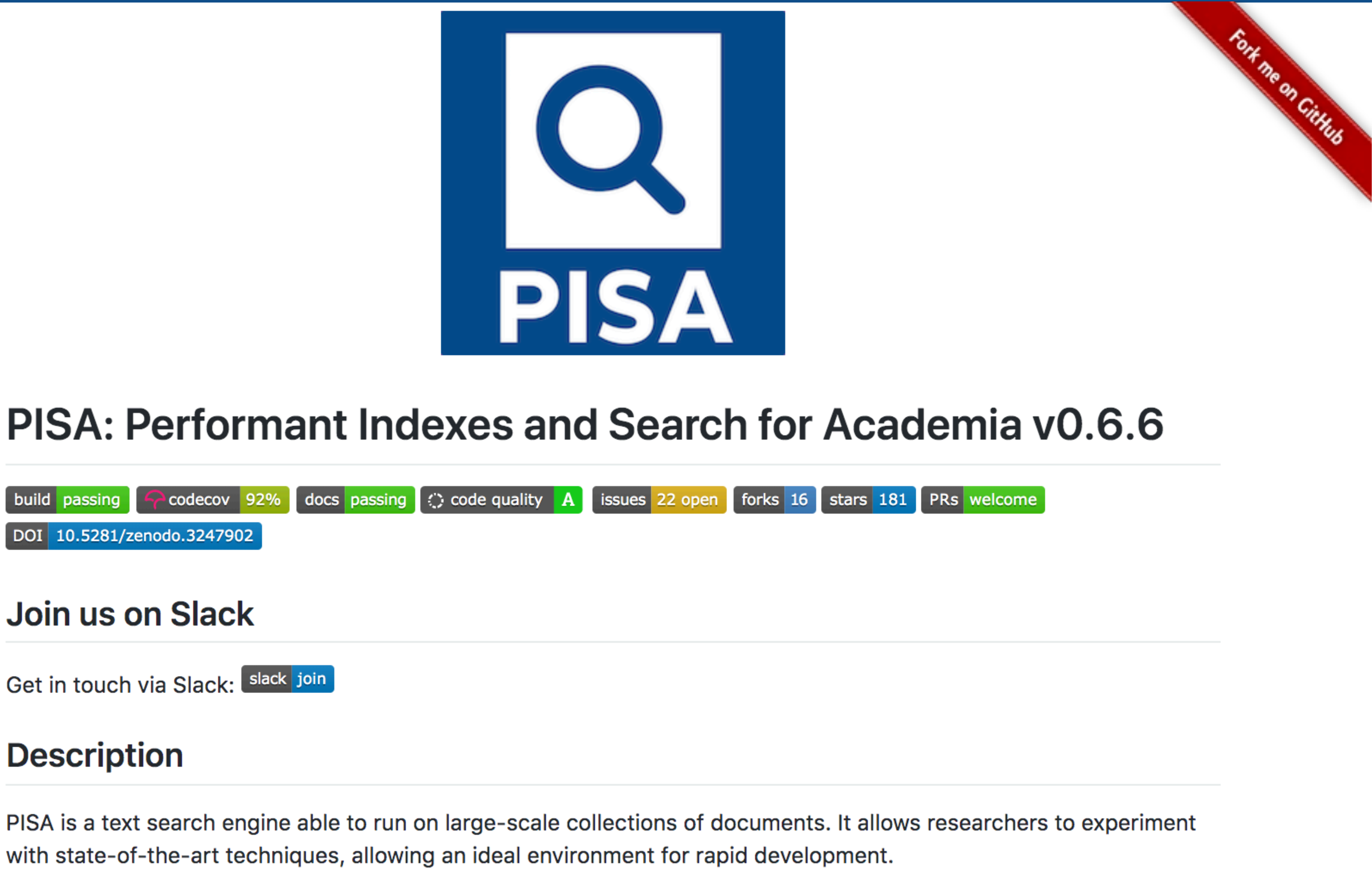
Joel Mackenzie
RMIT University

Torsten Suel
New York University

The Open-Source IR Replicability Challenge (OSIRRC 2019)

July 25, 2019

PISA: Performant Indexes and Search for Academia


Fork me on GitHub

PISA

PISA: Performant Indexes and Search for Academia v0.6.6

build passing | codecov 92% | docs passing | code quality A | issues 22 open | forks 16 | stars 181 | PRs welcome

DOI 10.5281/zenodo.3247902

Join us on Slack

Get in touch via Slack: slack join

Description

PISA is a text search engine able to run on large-scale collections of documents. It allows researchers to experiment with state-of-the-art techniques, allowing an ideal environment for rapid development.

github.com/pisa-engine/pisa/

Design Overview

PISA is designed to be efficient, extensible, and easy to use.

Modern C++17 implementation

Low level optimizations: CPU intrinsics, branch prediction hinting, and SIMD instructions

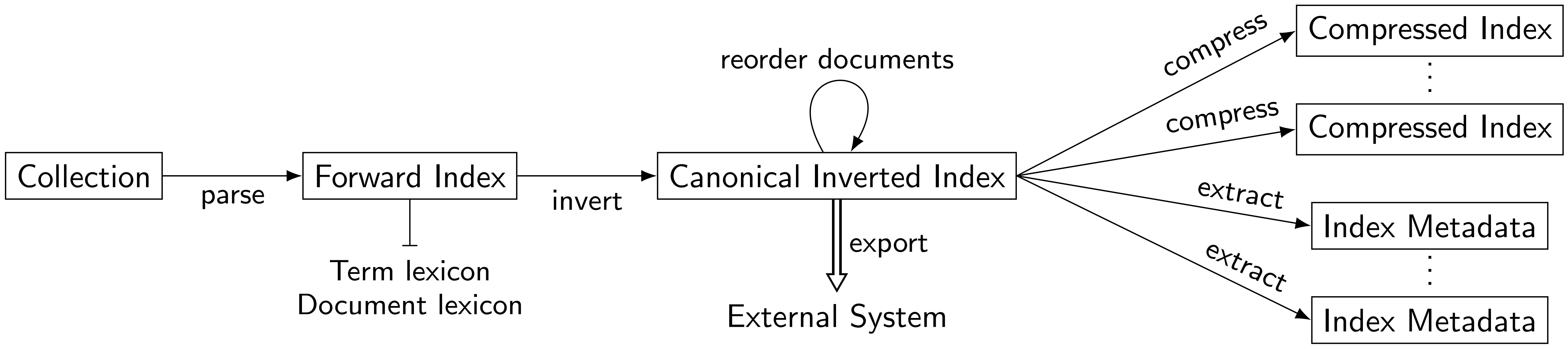
Extensible: pluggable parsers, stemmers, compression codecs, and query processign algorithms

Indexing, parsing and sharding capabilities

Implementation of document reordering

Free and open-source with permissive license

Index building pipeline



Parsing Collection

Several archive parsers, HTML content parser, tokenizer, and stemming algorithm.

Indexing

To produce an inverted index in the an uncompressed and universally readable format from a forward index

Document Reordering

To reassign the document identifiers within the inverted index: Random, URL, MinHash and BP.

Index Compression

Variable Byte encoders, word-aligned encoders, monotonic encoders, and frame-of-reference encoders.

Supported Collections

- **Robust04** consists of newswire articles from a variety of sources from the late 1980's through to the mid 1990's.Core17 - the New York Times corpus.
- **Core17** corresponds to the New York Times news collection, which contains news articles between 1987 and 2007.
- **Core18** is the TREC Washington Post Corpus, which consists of news articles and blog posts from January 2012 through August 2017.
- **Gov2** is the TREC 2004 Terabyte Track test collection consisting of around 25 million .gov sites crawled in early 2004; the documents are truncated to 256 kB.
- **ClueWeb09** is the ClueWeb 2009 Category B collection consisting of around 50 million English web pages crawled between January and February, 2009.
- **ClueWeb12** is the ClueWeb 2012 Category B-13 collection, which contains around 52 million English web pages crawled between February and May, 2012.

Feature Overview

- **Scoring**: BM25, Language Models, DPH, PL2
- **Search**: Boolean and scored conjunctions or disjunctions
- **Traversal strategy**: Document-at-a-Time or Term-at-a-Time
- **Dynamic pruning algorithms**: MaxScore and WAND, and their Block-Max counterparts, Block-Max MaxScore (BMM) and Block-Max WAND (BMW)
- **Variable-sized blocks** can be built (in lieu of fixed-sized blocks) to support the variable-block family of BlockMax algorithms, such as Variable Block-Max WAND (VBMW)

System Effectiveness

We process rank-safe, disjunctive, top-k queries to depth k = 1,000

	Topics	MAP	P@30	NDCG@20
Robust04	All	0.2534	0.3120	0.4221
Core17	All	0.2078	0.4260	0.3898
Core18	All	0.2384	0.3500	0.3927
Gov2	701-750	0.2638	0.4776	0.4070
	751-800	0.3305	0.5487	0.5073
	801-850	0.2950	0.4680	0.4925
ClueWeb09	51-100	0.1009	0.2521	0.1509
	101-150	0.1093	0.2507	0.2177
	151-200	0.1054	0.2100	0.1311
ClueWeb12	201-250	0.0449	0.1940	0.1529
	251-300	0.0217	0.1240	0.1484

Future Plans

PISA is still a relatively young project, aspiring to become a more widely used tool for IR experimentation.

Many relevant features can be still developed to further enrich the framework:

- Precomputed quantized partial scores
- Score-at-a-Time
- Learning-To-Rank (LTR)
- Query expansion
- Boilerplate removal
- Distributed indexes

Lesson Learned

- Docker is good for reproducibility
- Architecture-optimized binaries are not portable using Docker
- The collection format can still cause some issues
- Performance does not seem to be affected by the use of Docker

Thank you for your attention!

Any questions?