Whoosh

An open-source pure-Python search library

matt@whoosh.ca

Agenda

- Who am I?
- What is Whoosh?
- How does a inverted index work?
- Demo
- Advanced features
- What's next?

Matt Chaput

- Technical writer, Graphic designer, and UI designer
- Self-taught programmer -
  BASIC, Logo, Scheme, Smalltalk, JavaScript, Java, Python
- Information retrieval dilettante

Side Effects Software

- Houdini, a high-end 3D animation and special effects package
- Used in film, games, and commercials
- Twice recognized by the Academy of Motion Pictures, Arts and Sciences
- Uses Python as its scripting language

Before Whoosh

- Java Lucene in background process
- Customers HATED Java requirement
- Shipping Java was a big hassle with Sun

Why write a pure Python search library?

- Compiled libraries: installation problems and crashes, cross-platform issues
- Whoosh works where Python works
- I got 99 problems but make install ain't one

What happened was...

- Wrote a search library for use in Houdini help system
- Would this be useful to anyone else?
- Open sourced in 2007
- Front page of Hacker News
- Whoosh was pretty slow
- Now it's... much less slow ;)

Who uses Whoosh?

- Houdini online help
- MoinMoin
- Django-Haystack
- Lots of users on Bitbucket and mailing list

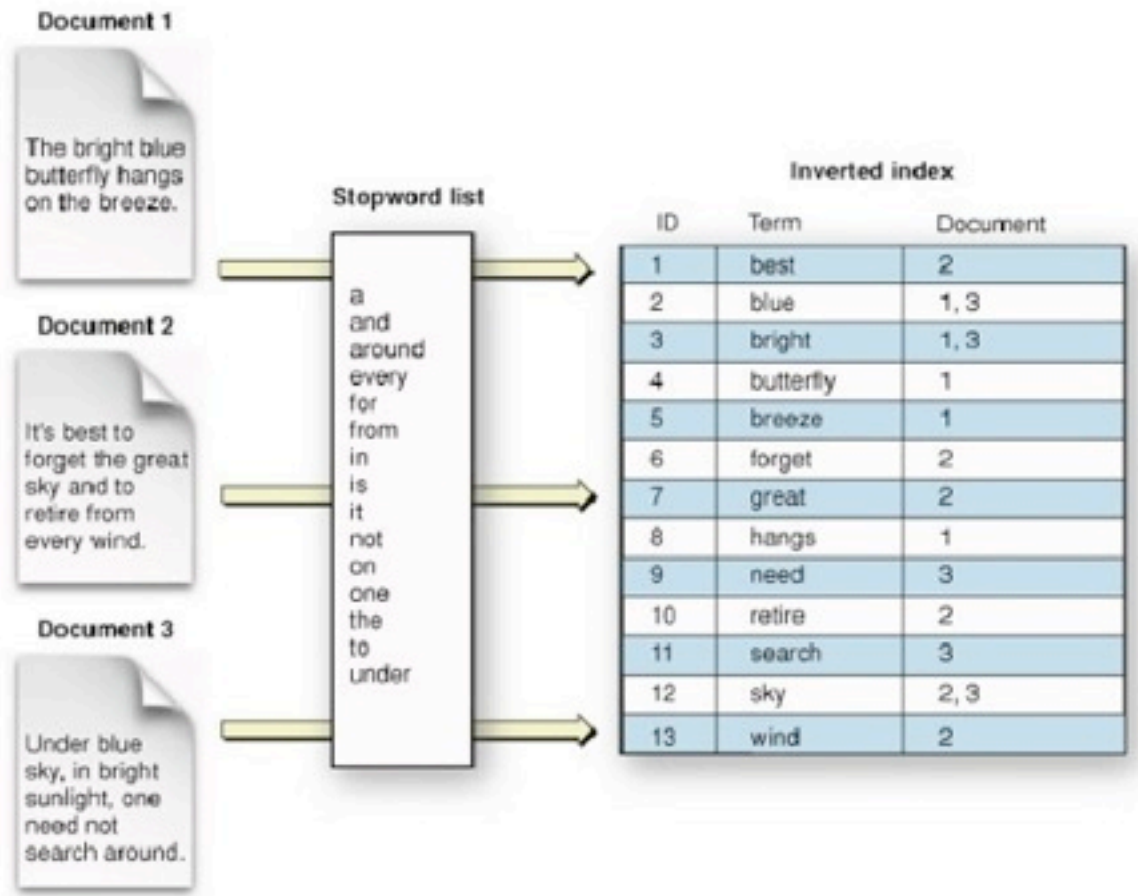
The basics

- Programming library
  - Toolkit for building your own search engine
- Not a web crawler (bring your own text)
- Thread and multiprocess safe
- Two-clause BSD license (GPL compatible)
- Python 2.5+, Python 3 compatible

Features

- Fields
- Text analysis
- Spell checker
- “More like this”
- Pluggable components
- Powerful queries
- Term Vectors
- Faceting and sorting
- Highlighted snippets
- Nested searching
- Efficient numeric and date fields
- Documentation & tests

Inverted Index



Source: developer.apple.com

Indexing

- Text analysis
  - Translate text into terms
- List of all postings in all documents
- External merge sort
  - Map of terms to posting lists

Searching

- Parse user query
- Run search
  - Grouping/sorting
  - Paging
- Spelling corrections
- Highlighted snippets
- “More like this”

Defining the schema

```
class MySchema(fields.SchemaClass):
    title = fields.TEXT(stored=True, sortable=True)
    content = fields.TEXT(spelling=True)
    path = fields.STORED
    modified = fields.DATETIME(stored=True, sortable=True)


# or


myschema = fields.Schema(
    title=fields.TEXT(stored=True, sortable=True),
    content=fields.TEXT(spelling=True),
    path=fields.STORED,
    modified=fields.DATETIME(stored=True, sortable=True),
)
```

Indexing

```
from whoosh import fields, index

class MySchema(fields.SchemaClass):
    title = fields.TEXT(stored=True, sortable=True)
    content = fields.TEXT
    indexed_on = fields.DATETIME(stored=True, sortable=True)
    summary = fields.STORED


ix = index.create_in("indexdir")
with ix,writer() as w:
    w.add_document(title="PyCon 2013",
                   content=myfile.read(),
                   indexed_on=datetime.now(),
                   summary="Conference report.")
```

Searching

```
from whoosh import index, qparser

ix = index.open_dir("indexdir")

qp = qparser.QueryParser("content", ix.schema)
q = qp.parse("guido")

with ix.searcher() as s:
    results = s.search(q)
    for hit in results:
        print(hit["title"])
```

Query types

- Term
- And, Or, Not
- DisjunctionMax
- Nested
- Phrase
- Near, Contains, etc.
- Range
- Prefix
- Wildcard
- Regex
- Fuzzy

Demo

Advanced features

- Multiprocessing for faster indexing
- Sorting/grouping
- Custom collectors
  (e.g. which queries matched?)
- Hierarchical searching
- Custom column types
- Codecs

Pure Python advantages

- Fun!
- No compilation - just works
- Fast iteration as you design the index
- Start up an interpreter and inspect the index, try queries, etc.
- Easy integration with other Python code

Pain points

- Performance
  - Try to touch as much C as possible
  - Do some silly things (full binary numbers, cPickle) because they're fast
- Python 3 transition (u, bytes, str)
- Dynamic typing in the large

Future directions

- Explore manipulating byte array slices instead of strings
- Keep more in RAM
- Experimental codecs
- More choice between fast and compact index
- Split out sub-systems

Interesting code

- "StructFile" class
- Fast on-disk hash table
- Flexible date parser
- Text analyzers
- On-disk FSA/FST
- Cross-platform file lock
- Space-efficient integer sets
- On-disk columnar storage

Resources

- Bitbucket repo: https://bitbucket.org/mchaput/whoosh
- Documentation: https://whoosh.readthedocs.org/en/latest/
- Mailing list: http://groups.google.com/group/whoosh
- matt@whoosh.ca

Thank you!