Bleve

Text Indexing for Go
1 February 2015

Marty Schoch

Say What?

blev-ee

bih-leev

Marty Schoch

- NoSQL Document Database
- Official Go SDK

Projects Using Go

- N1QL Query Language
- Secondary Indexing
- Cross Data-Center Replication

Why?

elasticsearch.

Apache Solr

Lucene

- Lucene/Solr/Elasticsearch are awesome
- Could we build 50% of Lucene's text analysis, combine it with off-the-shelf KV stores and get something interesting?

Bleve Core Ideas

Text Analysis Pipeline

- We only have to build common core
- Users customize for domain/language through interfaces

Pluggable KV storage

- No custom file format
- Plug-in Bolt, LevelDB, ForestDB, etc

Search

- Make term search work
- Almost everything else built on top of that...

What is Search?

Simple Search

Advanced Search

Google

Advanced Search

Find pages with...

all these words:	golang
this exact word or phrase:	
any of these words:	
none of these words:	
numbers ranging from:	to

Search Results

About 4,750 results (0.87 seconds)

Did you mean: ***bleve search***

Spelling Suggestions

blevesearch/bleve · GitHub
https://github.com/**blevesearch**/bleve
A modern text indexing library for go. Contribute to bleve development by creating an account on GitHub.

Result Text Snippets

bleve - modern text indexing for Go
www.**blevesearch**.com/
import "github.com/**blevesearch**/bleve" func main() { // open a new index mapping := bleve.NewIndexMapping() index, [illegible] New("example.bleve" ...

Highlighted Search Terms

bleve (@blevesearch) | Twitter
https://twitter.com/**blevesearch**
The latest Tweets from bleve (@**blevesearch**). modern text indexing for go.

Faceted Search

1-12 of 15 results for **Books** : "golang"

Show results for

- Any Category
- **Books**
  - Programming (11)
  - Computer Programming Language & Tool (9)
  - Reference (10)
  - Introductory & Beginning Programming (2)
  - Software Development (2)
  - Software (2)
  - Web Services (1)
  - Internet & Web Culture (2)
  - Software Utilities (1)
  - Linux Operating System (2)
  - Computers & Technology (12)
- + See more

Related Searches: go, go programming, haskell.

Book Format: Kindle Edition | Paperback

An Introduction to Programming in Go Sep 3, 2012
by Caleb Doxsey

Paperback
$10.00 Prime
Get it by **Thursday, Jan 22**

More Buying Choices
$5.78 used & new (6 offers)

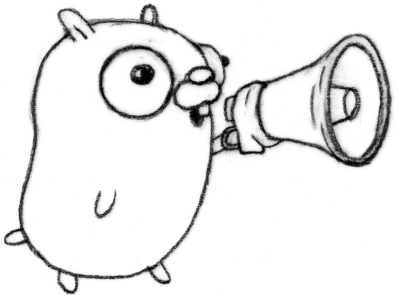
Kindle Edition
$3.00
Auto-delivered wirelessly

Go: Up and Running Apr 25, 2015
by Alan Harris

Paperback

Getting Started

Install bleve



go get github.com/blevesearch/bleve/...

Import

```
10 import "github.com/blevesearch/bleve"
11
12 type Person struct {
13     Name string
14 }
15
16 func main() {
17     mapping := bleve.NewIndexMapping()
18     index, err := bleve.New("people.bleve", mapping)
19     if err != nil {
20         log.Fatal(err)
21     }
22
23     person := Person{"Marty Schoch"}
24     err = index.Index("m1", person)
25     if err != nil {
26         log.Fatal(err)
27     }
28     fmt.Println("Indexed Document")
29 }
```

Data Model

```go
import "github.com/blevesearch/bleve"

type Person struct {
    Name string
}

func main() {

    mapping := bleve.NewIndexMapping()
    index, err := bleve.New("people.bleve", mapping)
    if err != nil {
        log.Fatal(err)
    }

    person := Person{"Marty Schoch"}
    err = index.Index("m1", person)
    if err != nil {
        log.Fatal(err)
    }
    fmt.Println("Indexed Document")
}
```

Index Mapping

```go
import "github.com/blevesearch/bleve"

type Person struct {
    Name string
}

func main() {
    mapping := bleve.NewIndexMapping()
    index, err := bleve.New("people.bleve", mapping)
    if err != nil {
        log.Fatal(err)
    }

    person := Person{"Marty Schoch"}
    err = index.Index("m1", person)
    if err != nil {
        log.Fatal(err)
    }
    fmt.Println("Indexed Document")
}
```

Create a New Index

```
10 import "github.com/blevesearch/bleve"
11
12 type Person struct {
13     Name string
14 }
15
16 func main() {
17     mapping := bleve.NewIndexMapping()
18     index, err := bleve.New("people.bleve", mapping)
19     if err != nil {
20         log.Fatal(err)
21     }
22
23     person := Person{"Marty Schoch"}
24     err = index.Index("m1", person)
25     if err != nil {
26         log.Fatal(err)
27     }
28     fmt.Println("Indexed Document")
29 }
```

Index Data

```go
import "github.com/blevesearch/bleve"

type Person struct {
    Name string
}

func main() {
    mapping := bleve.NewIndexMapping()
    index, err := bleve.New("people.bleve", mapping)
    if err != nil {
        log.Fatal(err)
    }

    person := Person{"Marty Schoch"}
    err = index.Index("m1", person)
    if err != nil {
        log.Fatal(err)
    }
    fmt.Println("Indexed Document")
}
```

Run

Open Index

```
15 func main() {
16     index, err := bleve.Open("people.bleve")
17     if err != nil {
18         log.Fatal(err)
19     }
20
21     query := bleve.NewTermQuery("marty")
22     request := bleve.NewSearchRequest(query)
23     result, err := index.Search(request)
24     if err != nil {
25         log.Fatal(err)
26     }
27     fmt.Println(result)
28 }
```

Build Query

```
15 func main() {
16     index, err := bleve.Open("people.bleve")
17     if err != nil {
18         log.Fatal(err)
19     }
20
21     query := bleve.NewTermQuery("marty")
22     request := bleve.NewSearchRequest(query)
23     result, err := index.Search(request)
24     if err != nil {
25         log.Fatal(err)
26     }
27     fmt.Println(result)
28 }
```

Build Request

```
15 func main() {
16     index, err := bleve.Open("people.bleve")
17     if err != nil {
18         log.Fatal(err)
19     }
20
21     query := bleve.NewTermQuery("marty")
22     request := bleve.NewSearchRequest(query)
23     result, err := index.Search(request)
24     if err != nil {
25         log.Fatal(err)
26     }
27     fmt.Println(result)
28 }
```

Search

```
15 func main() {
16     index, err := bleve.Open("people.bleve")
17     if err != nil {
18         log.Fatal(err)
19     }
20
21     query := bleve.NewTermQuery("marty")
22     request := bleve.NewSearchRequest(query)
23     result, err := index.Search(request)
24     if err != nil {
25         log.Fatal(err)
26     }
27     fmt.Println(result)
28 }
```

Run

More Realistic Examples

FOSDEM Schedule of Events (iCal)

```
BEGIN:VEVENT
METHOD:PUBLISH
UID:2839@FOSDEM15@fosdem.org
TZID:Europe-Brussels
DTSTART:20150201T140000
DTEND:20150201T144500
SUMMARY:bleve - text indexing for Go
DESCRIPTION: Nearly every application today has a search component.  But delivering high quality
search results requires a long list of text analysis and indexing techniques.  With the bleve lib
rary, we bring advanced text indexing and search to your Go applications.  In this talk we'll exa
mine how the bleve library brings powerful text indexing and search capabilities to Go applicatio
ns.
CLASS:PUBLIC
STATUS:CONFIRMED
CATEGORIES:Go
URL:https:/fosdem.org/2015/schedule/event/bleve/
LOCATION:K.3.401
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;CN="Marty Schoch":invalid:nomail
END:VEVENT
```

FOSDEM Event Data Structure

```
type Event struct {
    UID         string    `json:"uid"`
    Summary     string    `json:"summary"`
    Description string    `json:"description"`
    Speaker     string    `json:"speaker"`
    Location    string    `json:"location"`
    Category    string    `json:"category"`
    URL         string    `json:"url"`
    Start       time.Time `json:"start"`
    Duration    float64   `json:"duration"`
}
```

Index FOSDEM Events

```
36      count := 0
37      batch := bleve.NewBatch()
38      for event := range parseEvents() {
39          batch.Index(event.UID, event)
40          if batch.Size() > 100 {
41              err := index.Batch(batch)
42              if err != nil {
43                  log.Fatal(err)
44              }
45              count += batch.Size()
46              batch = bleve.NewBatch()
47          }
48      }
49      if batch.Size() > 0 {
50          index.Batch(batch)
51          if err != nil {
52              log.Fatal(err)
53          }
54          count += batch.Size()
55      }
56      fmt.Printf("Indexed %d Events\n", count)
```

Run

Search FOSDEM Events

```
11 func main() {
12
13     index, err := bleve.Open("fosdem.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     q := bleve.NewTermQuery("bleve")
19     req := bleve.NewSearchRequest(q)
20     req.Highlight = bleve.NewHighlightWithStyle("html")
21     req.Fields = []string{"summary", "speaker"}
22     res, err := index.Search(req)
23     if err != nil {
24         log.Fatal(err)
25     }
26     fmt.Println(res)
27 }
```

Run

Phrase Search

```
11 func main() {
12
13     index, err := bleve.Open("fosdem.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     phrase := []string{"advanced", "text", "indexing"}
19     q := bleve.NewPhraseQuery(phrase, "description")
20     req := bleve.NewSearchRequest(q)
21     req.Highlight = bleve.NewHighlightWithStyle("html")
22     req.Fields = []string{"summary", "speaker"}
23     res, err := index.Search(req)
24     if err != nil {
25         log.Fatal(err)
26     }
27     fmt.Println(res)
28 }
```

Run

Combining Queries

```
11 func main() {
12
13     index, err := bleve.Open("fosdem.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     tq1 := bleve.NewTermQuery("text")
19     tq2 := bleve.NewTermQuery("search")
20     q := bleve.NewConjunctionQuery([]bleve.Query{tq1, tq2})
21     req := bleve.NewSearchRequest(q)
22     req.Highlight = bleve.NewHighlightWithStyle("html")
23     req.Fields = []string{"summary", "speaker"}
24     res, err := index.Search(req)
25     if err != nil {
26         log.Fatal(err)
27     }
28     fmt.Println(res)
29 }
```

Run

Combining More Queries

```
11 func main() {
12
13     index, err := bleve.Open("fosdem.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     tq1 := bleve.NewTermQuery("text")
19     tq2 := bleve.NewTermQuery("search")
20     tq3 := bleve.NewTermQuery("believe")
21     q := bleve.NewConjunctionQuery(
22         []bleve.Query{tq1, tq2, tq3})
23     req := bleve.NewSearchRequest(q)
24     req.Highlight = bleve.NewHighlightWithStyle("html")
25     req.Fields = []string{"summary", "speaker"}
26     res, err := index.Search(req)

27     if err != nil {
28         log.Fatal(err)
29     }
30     fmt.Println(res)
31 }
```

Run

Fuzzy Query

```
11 func main() {
12
13     index, err := bleve.Open("fosdem.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     tq1 := bleve.NewTermQuery("text")
19     tq2 := bleve.NewTermQuery("search")
20     tq3 := bleve.NewFuzzyQuery("believe")
21     q := bleve.NewConjunctionQuery(
22         []bleve.Query{tq1, tq2, tq3})
23     req := bleve.NewSearchRequest(q)
24     req.Highlight = bleve.NewHighlightWithStyle("html")
25     req.Fields = []string{"summary", "speaker"}

26     res, err := index.Search(req)
27     if err != nil {
28         log.Fatal(err)
29     }
30     fmt.Println(res)
31 }
```

Run

Numeric Range Query

```
11 func main() {
12
13     index, err := bleve.Open("fosdem.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     longTalk := 110.0
19     q := bleve.NewNumericRangeQuery(&longTalk, nil)
20     req := bleve.NewSearchRequest(q)
21     req.Highlight = bleve.NewHighlightWithStyle("html")
22     req.Fields = []string{"summary", "speaker", "duration"}
23     res, err := index.Search(req)
24     if err != nil {
25         log.Fatal(err)
26     }
27     fmt.Println(res)
28 }
```

Run

Date Range Query

```
11 func main() {
12
13     index, err := bleve.Open("fosdem.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     lateSunday := "2015-02-01T17:30:00Z"
19     q := bleve.NewDateRangeQuery(&lateSunday, nil)
20     q.SetField("start")
21     req := bleve.NewSearchRequest(q)
22     req.Highlight = bleve.NewHighlightWithStyle("html")
23     req.Fields = []string{"summary", "speaker", "start"}
24     res, err := index.Search(req)
25     if err != nil {
26         log.Fatal(err)
27     }
28     fmt.Println(res)
29 }
```

Run

Query Strings

```
11 func main() {
12
13     index, err := bleve.Open("fosdem.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     qString := `+description:text `
19     qString += `summary:"text indexing" `
20     qString += `summary:believe~2 `
21     qString += `-description:lucene `
22     qString += `duration:>30`
23     q := bleve.NewQueryStringQuery(qString)
24     req := bleve.NewSearchRequest(q)
25     req.Highlight = bleve.NewHighlightWithStyle("html")
26     req.Fields = []string{"summary", "speaker", "description", "duration"}
27     res, err := index.Search(req)
28     if err != nil {
29         log.Fatal(err)
30     }
31     fmt.Println(res)
32 }
```

Run

Default Mapping vs Custom Mapping

The default mapping has worked really well, but...

```
18      q := bleve.NewTermQuery("haystack")
19      req := bleve.NewSearchRequest(q)
20      req.Highlight = bleve.NewHighlightWithStyle("html")
21      req.Fields = []string{"summary", "speaker"}
22      res, err := index.Search(req)
23      if err != nil {
24          log.Fatal(err)
25      }

26      fmt.Println(res)
```

Run

Earlier today we heard talk named **"Finding Bad Needles in Worldwide Haystacks"**.

Will we find it if we search for **"haystack"**?

Custom Mapping

```
27      enFieldMapping := bleve.NewTextFieldMapping()
28      enFieldMapping.Analyzer = "en"
29
30      eventMapping := bleve.NewDocumentMapping()
31      eventMapping.AddFieldMappingsAt("summary", enFieldMapping)
32      eventMapping.AddFieldMappingsAt("description", enFieldMapping)
33
34      kwFieldMapping := bleve.NewTextFieldMapping()
35      kwFieldMapping.Analyzer = "keyword"
36
37      eventMapping.AddFieldMappingsAt("url", kwFieldMapping)
38      eventMapping.AddFieldMappingsAt("category", kwFieldMapping)
39
40      mapping := bleve.NewIndexMapping()
41      mapping.DefaultMapping = eventMapping
42
43      index, err := bleve.New("custom.bleve", mapping)
44      if err != nil {
45          log.Fatal(err)
46      }
```

Run

Search Custom Mapping

```
18      q := bleve.NewTermQuery("haystack")
19      req := bleve.NewSearchRequest(q)
20      req.Highlight = bleve.NewHighlightWithStyle("html")
21      req.Fields = []string{"summary", "speaker"}
22      res, err := index.Search(req)
23      if err != nil {
24          log.Fatal(err)
25      }
26      fmt.Println(res)
```

Run

Analysis Wizard

http://analysis.blevesearch.com

Watch a video introduction!

Bleve Text Analysis Wizard

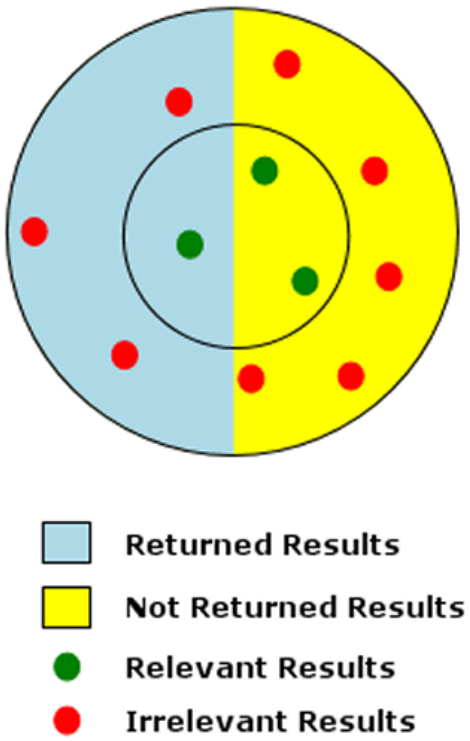
To get high quality search results, you must choose the right analyzer for your source text. To better understand how your text will be analyzed, try it out using the form below.

Analyzer en haystacks ▶ Analyze

Text: haystacks

Position	Term	Start	End
1	haystack	0	9

Precision vs Recall



- Precision - are the returned results relevant?
- Recall - are the relevant results returned?

Faceted Search

```
11 func main() {
12
13     index, err := bleve.Open("custom.bleve")
14     if err != nil {
15         log.Fatal(err)
16     }
17
18     q := bleve.NewMatchAllQuery()
19     req := bleve.NewSearchRequest(q)
20     req.Size = 0
21     req.AddFacet("categories",
22         bleve.NewFacetRequest("category", 50))
23     res, err := index.Search(req)
24     if err != nil {
25         log.Fatal(err)
26     }
27     fmt.Println(res)
28 }
```

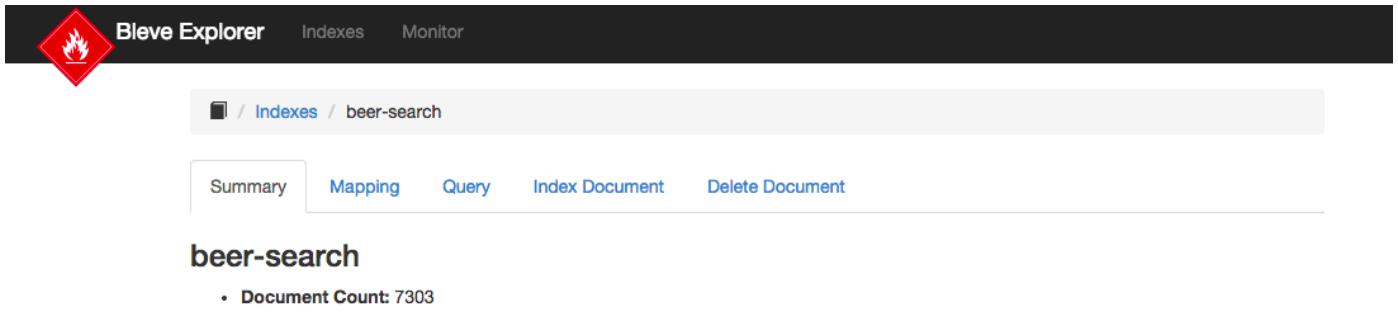
Run

Optional HTTP Handlers

```
import "github.com/blevesearch/bleve/http"
```

- All major bleve operations mapped
- Assume JSON document bodies
- See bleve-explorer sample app

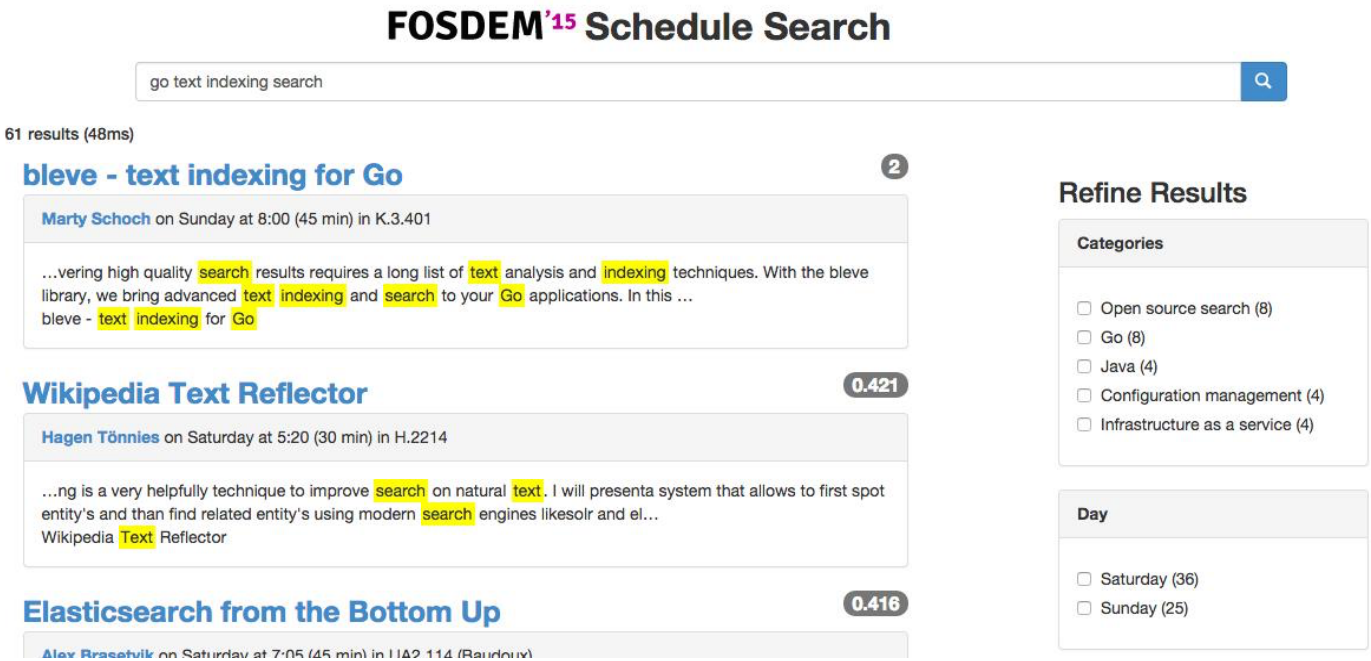
```
https://github.com/blevesearch/bleve-explorer
```



Putting it All Together

FOSDEM Schedule Search

http://fosdem.blevesearch.com



Performance

Micro Benchmarks

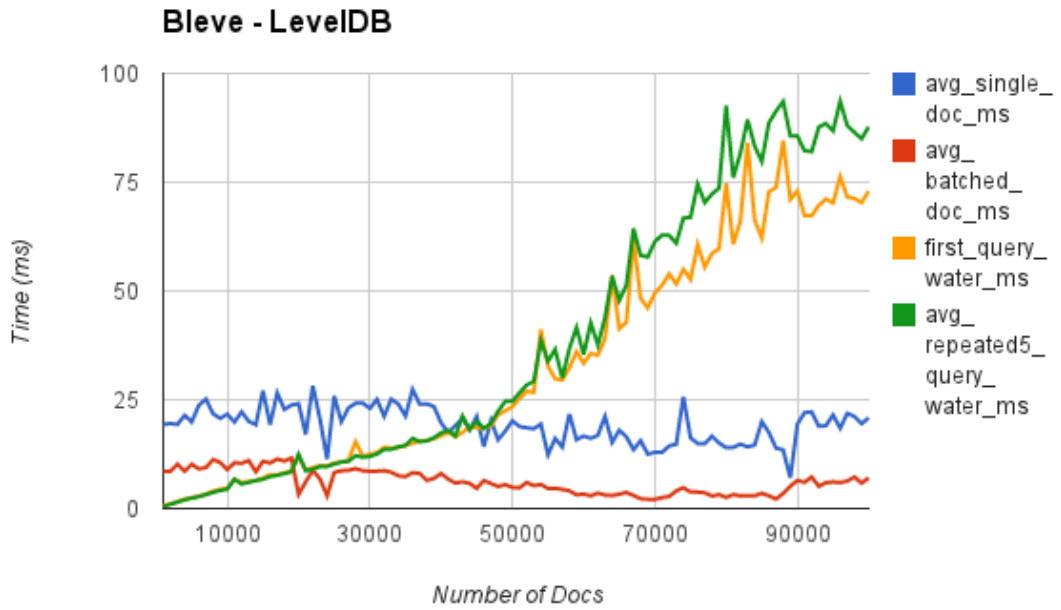
Use Go benchmarks to test/compare small units of functionality in isolation.

```
$ go test -bench=. -cpu=1,2,4
PASS
BenchmarkBoltDBIndexing1Workers        1000       3075988 ns/op
BenchmarkBoltDBIndexing1Workers-2        1000       4004125 ns/op
BenchmarkBoltDBIndexing1Workers-4         500       4470435 ns/op
BenchmarkBoltDBIndexing2Workers         500       3148049 ns/op
BenchmarkBoltDBIndexing2Workers-2        1000       3336268 ns/op
BenchmarkBoltDBIndexing2Workers-4        1000       3461157 ns/op
BenchmarkBoltDBIndexing4Workers        1000       3642691 ns/op
BenchmarkBoltDBIndexing4Workers-2         500       3130814 ns/op
BenchmarkBoltDBIndexing4Workers-4        1000       3312662 ns/op
BenchmarkBoltDBIndexing1Workers10Batch           1     1350916284 ns/op
BenchmarkBoltDBIndexing1Workers10Batch-2           1     1493538328 ns/op
BenchmarkBoltDBIndexing1Workers10Batch-4           1     1256294099 ns/op
BenchmarkBoltDBIndexing2Workers10Batch           1     1393491792 ns/op
BenchmarkBoltDBIndexing2Workers10Batch-2           1     1271605176 ns/op
BenchmarkBoltDBIndexing2Workers10Batch-4           1     1343410709 ns/op
BenchmarkBoltDBIndexing4Workers10Batch           1     1393552247 ns/op
BenchmarkBoltDBIndexing4Workers10Batch-2           1     1144501920 ns/op
BenchmarkBoltDBIndexing4Workers10Batch-4           1     1311805564 ns/op
BenchmarkBoltDBIndexing1Workers100Batch           3      425731147 ns/op
```

Bleve Bench

Long(er) running test, index real text from Wikipedia. Measure stats periodicaly, compare across time.

- Does indexing performance degrade over time?
- How does search performance relate to number of matching documents?



Join the Community

Community

freenode

- #bleve is small/quiet room, talk to us real time

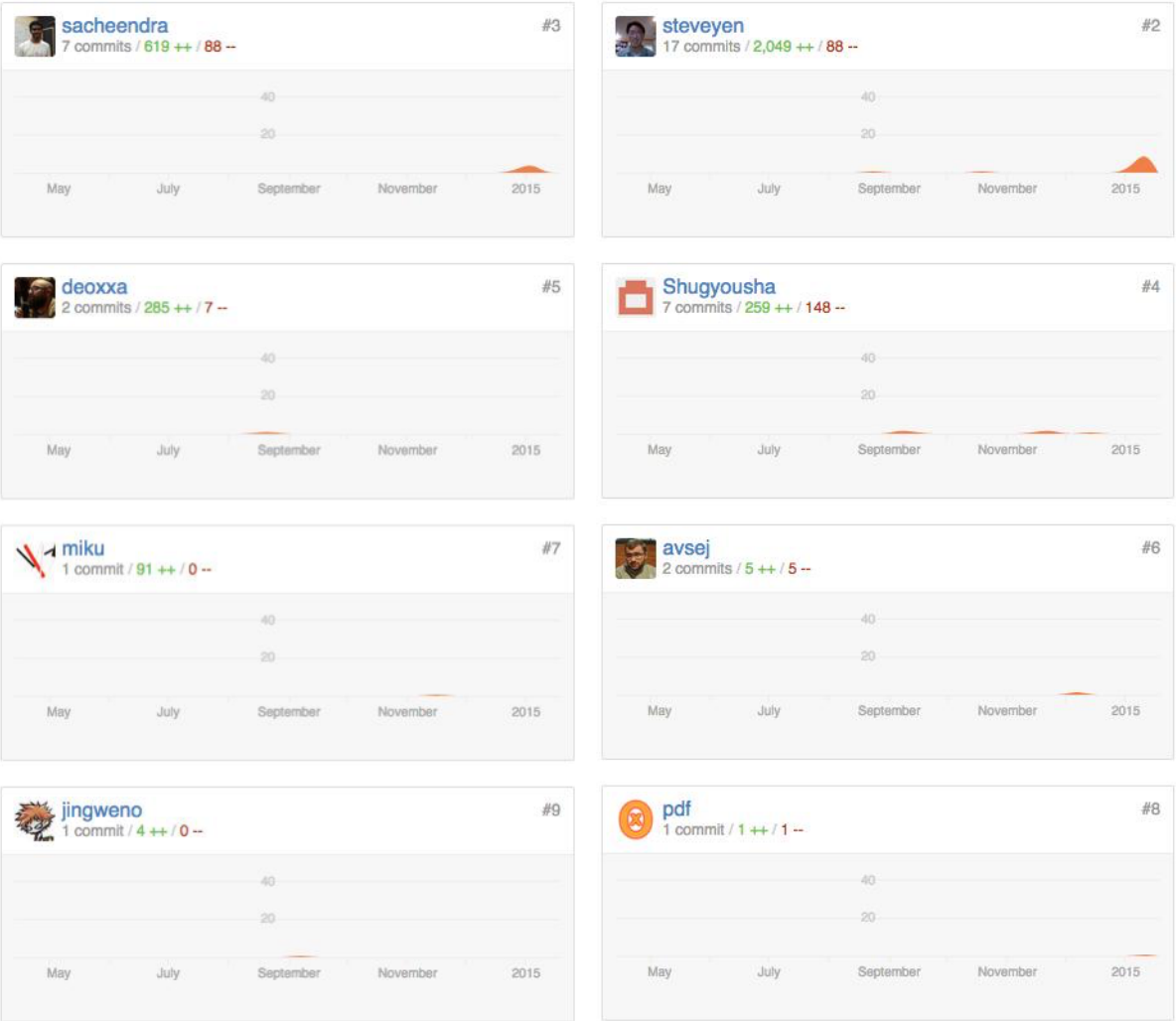
Google groups

- Discuss your use-case
- Plan a feature implementation

GitHub

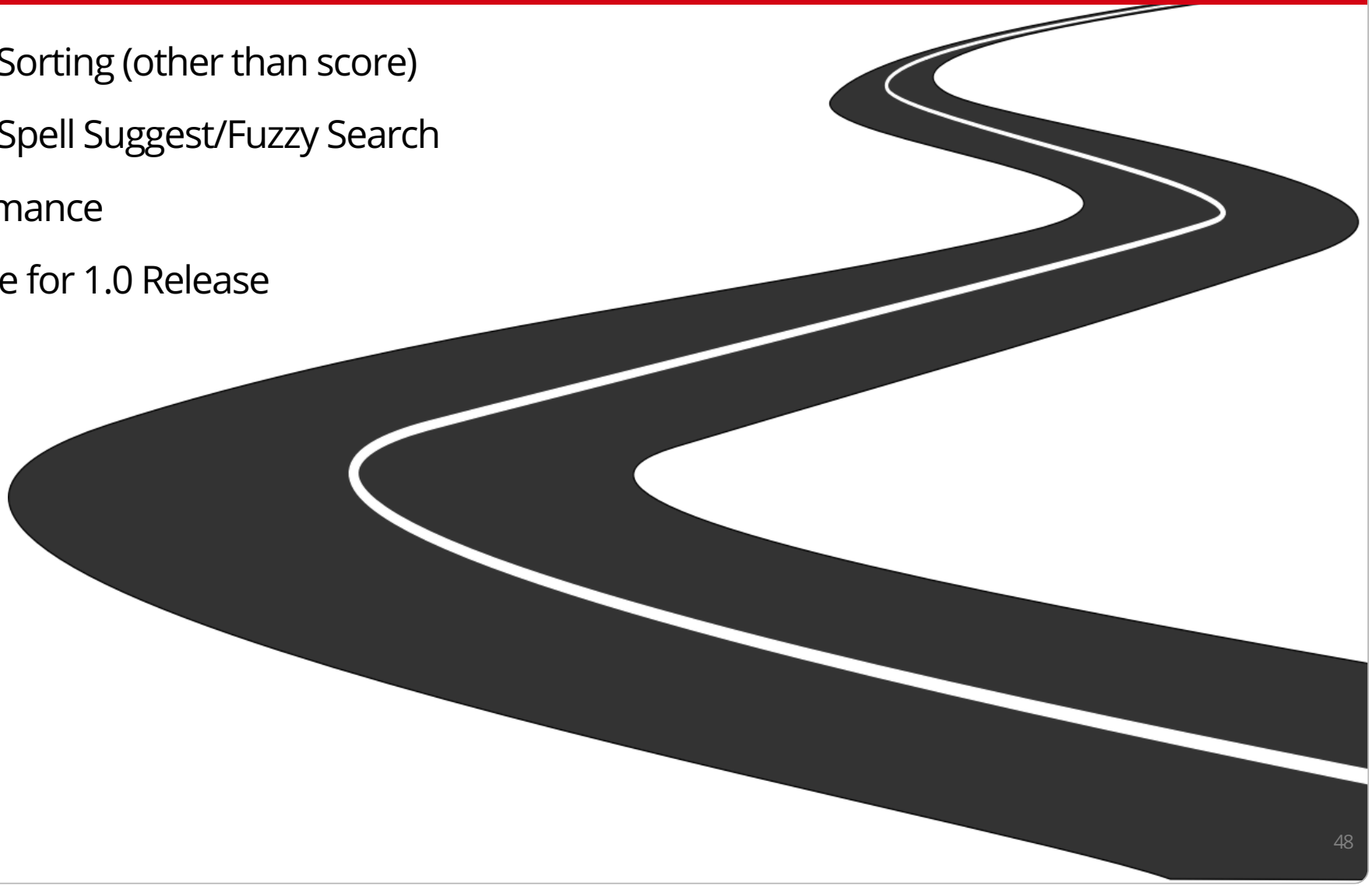
- Apache License v2.0, Report Issues, Submit Pull Requests

Contributors



Roadmap

- Result Sorting (other than score)
- Better Spell Suggest/Fuzzy Search
- Performance
- Prepare for 1.0 Release



Speaking

- GopherCon India February 2015 (Speaking)
- GopherCon July (Attending/Proposal to be Submitted)
- Your Conference/Meetup Here!

Thank you

Marty Schoch
marty@couchbase.com (mailto:marty@couchbase.com)
http://github.com/blevesearch/bleve (http://github.com/blevesearch/bleve)
@mschoch (http://twitter.com/mschoch)
@blevesearch (http://twitter.com/blevesearch)