THE MINION SEARCH ENGINE: INDEXING, SEARCH, TEXT SIMILARITY AND TAG GARDENING

Jeff Alexander, Member of Technical Staff, Sun Microsystems
Stephen Green, Senior Staff Engineer, Sun Microsystems

TS-5027

GOAL

Learn how the Minion Search Engine can be used to index and retrieve documents and how its advanced features allow you to find similar documents and cluster result sets. Learn how Minion can be used to build a powerful social tagging application.

Agenda

- Background: What Is Minion?
- Minion and Lucene
- Documents and Fields
- Indexing and Querying
- Tools for the Advanced Beginner
- Tag Based Recommendation
- Tag Gardening

Background

- Started as part of Sun™ Labs' Knowledge Technology Group
  - Sun Labs has had a search project since 1993
- Current version developed as part of the Advanced Search Technologies Project in Sun Labs
- A previous version of Minion shipped with the Java Enterprise System Portal Server and Web Server

Why Minion?

- Research platform
  - Passage retrieval
  - Text similarity
  - Document Classification
  - Results clustering
- Intended for Real World use
- Highly configurable
  - Configure: Indexing pipeline, tokenizers, postings types, dictionaries, weighting functions, classification algorithms, clustering algorithms

Minion: Features

- Concurrent indexing and retrieval
- Easy to use API for application development
- Field-centric approach to documents
- Ranked boolean, proximity, parametric query operators
- Multiple query languages
- Light weight and full morphology
- Find similar documents
- Results clustering
- Automatic document classification
- Word sense disambiguation

Agenda

- Background: What Is Minion?
- Minion and Lucene
- Documents and Fields
- Indexing and Querying
- Tools for the Advanced Beginner
- Tag Based Recommendation
- Tag Gardening

What about Lucene?

- A high quality open source Java search engine with strong developer and user communities
- The engines have much in common
  - Inverted file index with compressed postings
- Out-of-the-box configuration differs substantially
- Minion provides capabilities not currently available in Lucene
- Performance
  - Indexing is faster in Minion
  - Query performance is comparable

Minion vs. Lucene: Feature Comparison

- Fields
  - Minion
    - When querying, search for terms in all fields
    - Provides basic field types
  - Lucene
    - When querying, searches a single field
    - Provides only string field types
- Morphological variations
  - Minion
    - Index terms in documents as they appear
    - Expand query terms with known variations
  - Lucene
    - Index terms in documents as they appear
    - Query only for provided form

Minion vs. Lucene: Feature Comparison

- Case sensitivity
  - Minion
    - Indexes are case-sensitive
    - Querying is case-insensitive by default
    - Query operators can modify default behavior
  - Lucene
    - Indexes are case-insensitive
    - Case-sensitive queries enabled by indexing multiple case variations
- Well-defined API
  - Minion
    - Single interface package for indexing and retrieval capabilities
    - Designed to be all you need to know to use the engine
    - Runtime configuration via dependency injection
  - Lucene
    - Loosely defined set of API classes
    - Configuration via compilation

Minion: Features

- Concurrent indexing and retrieval
- Easy to use API for application development
- Field-centric approach to documents
- Ranked boolean, proximity, parametric query operators
- Multiple query languages
- Light weight and full morphology
- Find similar documents
- Results clustering
- Automatic document classification
- Word sense disambiguation

Agenda

- Background: What Is Minion?
- Minion and Lucene
- Documents and Fields
- Indexing and Querying
- Tools for the Advanced Beginner
- Tag Based Recommendation
- Tag Gardening

A Sample Document

```
From:        Some.Body@Example.COM (Some Body)
Date:        Fri, 12 Oct 2007 17:51:28 +0530
Subject:     [zfs-discuss] XFS_IOC_FSGETXATTR &
             XFS_IOC_RESVSP64 like options in ZFS?
Message-ID:  <470F66C8.3070704@example.com>

I am using XFS_IOC_FSGETXATTR in ioctl() call on Linux
running XFS file system. I want to use similar thing on
Solaris running ZFS file system.
```

Minion's Document Model

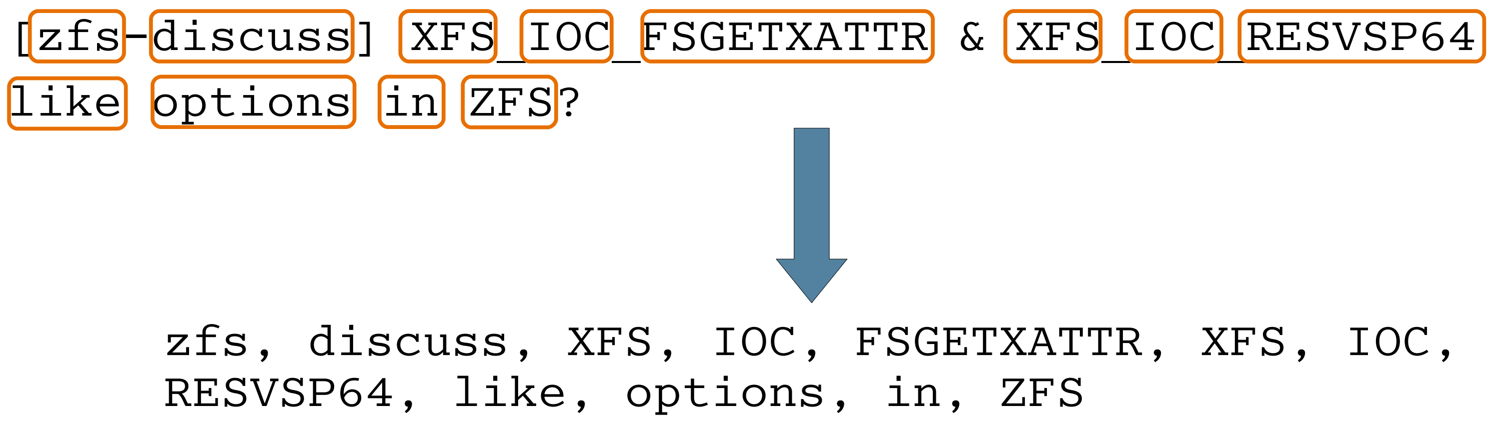
- Minion models a document as a map from field names to field values
  - Documents have a unique key
  - When you index a document with the same key the old data is replaced
- What are the fields in my documents going to be?
  - from, subject, sent-date, references, body
- What will I want to do with these fields when querying?
  - Look for words in the subject or body
  - Query by date range
  - Search for email sent from a particular domain

A Word Or Two About Fields

- Fields can have a number of attributes that determine
  - How the fields are treated at indexing time
  - What type of queries you can perform
- Fields can be
  - Indexed
  - Tokenized
  - Vectored
  - Saved

Indexed And Tokenized Fields

- Words in indexed fields
  - Are placed into the main dictionary
  - Can be used as terms in simple or fielded queries
- Tokenized fields are broken into words using a tokenizer
- The default tokenizer
  - Breaks text into words at whitespace and punctuation
  - Tokenizes non-space separated languages into overlapping bigrams



```
[zfs-discuss] XFS_IOC_FSGETXATTR & XFS_IOC_RESVSP64
like options in ZFS?
```

```
zfs, discuss, XFS, IOC, FSGETXATTR, XFS, IOC,
RESVSP64, like, options, in, ZFS
```

Vectored Fields

- Words in vectored fields
  - Are available for clustering, document similarity, and classification operations
  - Can be retrieved quickly for a given document

Term	Count
discuss	1
fsgetxattr	1
in	1
ioc	2
like	1

Term	Count
options	1
resvsp64	1
xfs	2
zfs	2

Saved Fields

- Saved fields
  - Store a copy of the field value in the index
  - Have a type
  - Can be retrieved from search results
  - Can be used for relational queries
  - Can be used to sort or group search results
- Multiple values can be stored for the same field

```
[zfs-discuss] XFS_IOC_FSGETXATTR & XFS_IOC_RESVSP64
like options in ZFS?
```

Agenda

- Background: What Is Minion?
- Minion and Lucene
- Documents and Fields
- Indexing and Querying
- Tools for the Advanced Beginner
- Tag Based Recommendation
- Tag Gardening

Indexing with Minion: Getting An Engine

```
SearchEngine engine =
    SearchEngineFactory.getSearchEngine(indexDir);
```

- This uses the default configuration for the engine
  - Case sensitive dictionaries, word positions stored in postings, and field information stored in postings
- You can specify a runtime configuration file for
  - Indexing pipeline
  - Case sensitivity
  - Query timeouts
  - Morphology and stemming
  - Caching behavior
  - Term weighting functions
  - Classification and clustering algorithms

Defining Fields

```
// Get attributes for an indexed, tokenized, vectored field
EnumSet<FieldInfo.Attribute> ia =
    FieldInfo.getIndexedAttributes();
EnumSet<FieldInfo.Attribute> isa = ia.clone();
sa.add(FieldInfo.Attribute.SAVED);
EnumSet<FieldInfo.Attribute> sa =
    EnumSet.of(FieldInfo.Attribute.SAVED);

// Fields can be defined any time
engine.defineField(new FieldInfo("from", sa,
                                 FieldInfo.Type.STRING));
engine.defineField(new FieldInfo("subject", isa,
                                 FieldInfo.Type.STRING));
engine.defineField(new FieldInfo("sent-date", sa,
                                 FieldInfo.Type.DATE));
[...]
engine.defineField(new FieldInfo("message-num", sa,
                                 FieldInfo.Type.INTEGER));
engine.defineField(new FieldInfo("body", ia));
```

Indexing Documents

```
public void index(SimpleIndexer si, MimeMessage m,
                  String folderName) {
   si.startDocument(m.getMessageID());
   si.addField("from", m.getFrom()[0].toString());
   si.addField("subject", m.getSubject());
   si.addField("sent-date", m.getSentDate());
   if (m.getHeader("references") != null) {
     si.addField("references",
                 m.getHeader("references"));
   }
   [...]
   BufferedReader reader = new BufferedReader(...);
   String line = null;
   while ((line = reader.readLine()) != null) {
       si.addField("body", line);
   }
   si.endDocument();
}
```

Querying

```
ResultSet rs = engine.search(queryStr,
                              "-sent-date,+subject");
List<Result> results = rs.getResults(0, n);
System.out.printf("Displaying %d of %d results\n",
                  Math.min(n, rs.size()), rs.size());
for(Result r : results) {
  System.out.printf("%.3f %15s %d %tT %20s %s %s\n",
    r.getScore(),
    r.getSingleFieldValue("folder-name"),
    r.getSingleFieldValue("message-num"),
    r.getSingleFieldValue("sent-date"),
    r.getSingleFieldValue("from"),
    r.getField("references"),
    r.getSingleFieldValue("subject"));
}
```

Query Operators

- Term
  - `case`, `exact`, `morph`, `stem`, `wildcard`
  - Can be used on field values that have been indexed
- Proximity
  - `near`, `within`, `phrase`, `passage`
  - Can be used on field values that have been indexed
  - Can be restricted to a field using the `contains` operator
- Parametric
  - `=`, `!=`, `<`, `<=`, `>`, `>=`, `undefined`, `substring`, `starts`, `ends`, `similar`
  - Can be used on field values that have been saved
- Weighted Boolean
  - `and`, `or`, `not`, `weight`
  - Can be used to combine expressions of any kind

Available Grammars

- Full
  - `(subject <contains> (xfs <not> iscsi)) <and> (sent-date >= 10/01/07) fsgetxattr in ioctl on linux`
- Web
  - `+xfs "ioc_fsgetxattr" ioctl linux -iscsi`
- Lucenesque
  - `subject:xfs AND sent-date:[20071007 TO 20380118]`

Agenda

- Background: What Is Minion?
- Minion and Lucene
- Documents and Fields
- Indexing and Querying
- Tools for the Advanced Beginner
- Tag Based Recommendation
- Tag Gardening

Document Vectors

- A weighted vector of terms from the document
  - The weight of a term in a document indicates the importance of that term in that document
- Terms are taken from vectored fields
  - Any vectored field can be used to generate a document vector
  - An implicit vectored field includes the terms from all vectored fields

Term	Weight
discuss	0.000
fsgetxattr	0.080
in	0.004
ioc	0.114
like	0.015

Term	Weight
options	0.043
resvsp64	0.087
xfs	0.104
zfs	0.000

Java

JavaOne

Finding Similar Documents

```
Result r = anInterestingResult;
// Find similar based on whole document
ResultSet s = r.getDocumentVector().findSimilar();
// Find similar based on subject
ResultSet s =
r.getDocumentVector("subject").findSimilar();
// Find similar based on field combination
WeightedField[] fields = new WeightedField[] {
    new WeightedField("subject", 0.25),
    new WeightedField("body", 0.75)
};
ResultSet s = r.getDocumentVector(fields).findSimilar()
```

- Computes the cosine similarity between document vectors
- Speed things up by restricting the set of terms to a percentage of the most important terms in the vector

Document Classification

- Given a set of documents with a label, learn how to apply the label to new documents

```
ResultSet r = engine.search("to = zfs-discuss");
engine.trainClass(r, "zfs-discuss",
                  "related-to", "body");
```

- Once a classifier is trained, it is applied against all new items
- Minion can associate a confidence score with the assigned labels
- Documents can be assigned to zero or more classes

Clustering Documents

```
ResultSet rs = engine.search(queryStr, sortSpec);
Set<ResultsCluster> clusters = rs.cluster("subject", 4);
for(ResultsCluster c : clusters) {
    System.out.println(c.getDescription(6));
    displayResults(c.getResults(), 3);
}
```

- You can cluster based on any vectored field
- The `ResultsCluster` interface provides
    - A way to get a description of the cluster
    - A way to get the document closest to the center of the cluster
    - A way to find out how close a given document is to the cluster

Agenda

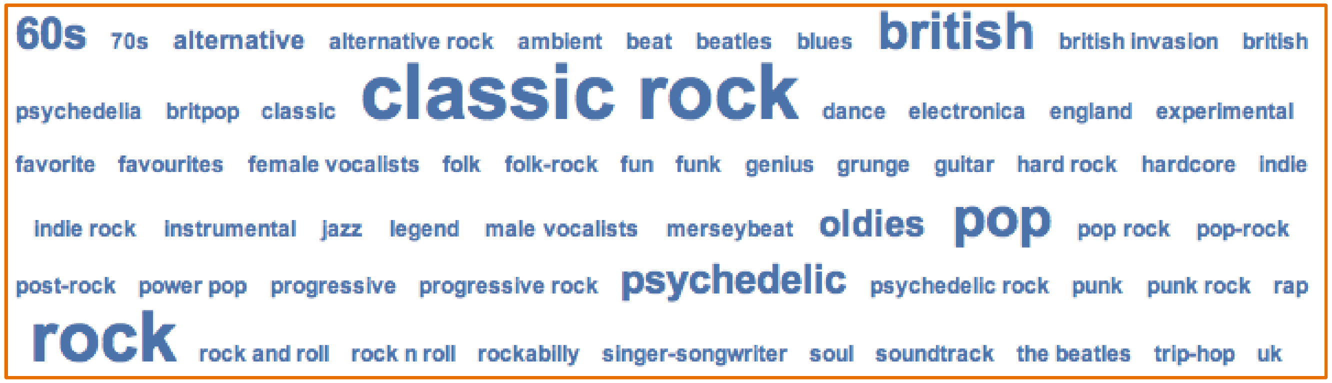
- Background: What Is Minion?
- Minion and Lucene
- Documents and Fields
- Indexing and Querying
- Tools for the Advanced Beginner
- Tag Based Recommendation
- Tag Gardening

What Are Search Engines Good At?

- Take a collection of documents that contain words
- For each document
  - Break out and count the words
  - Determine what the most important words are in the document
- For the collection
  - Count how often each term occurs
- Given a query
  - Find the set of documents containing the query terms
  - Rank the documents according to the importance of the query terms in the documents
- Given a document
  - Find documents that contain similar important terms

Not Your Father's Search Problem

- Say we have a set of social tags that have been applied to musical artists


60s 70s alternative alternative rock ambient beat beatles blues british british invasion british psychedelia britpop classic classic rock dance electronica england experimental favorite favourites female vocalists folk folk-rock fun funk genius grunge guitar hard rock hardcore indie indie rock instrumental jazz legend male vocalists merseybeat oldies pop pop rock pop-rock post-rock power pop progressive progressive rock psychedelic psychedelic rock punk punk rock rap rock rock and roll rock n roll rockabilly singer-songwriter soul soundtrack the beatles trip-hop uk

  - Data from Last.fm: 120,000 unique tags and 248,000 artists
- Given an artist, can we find similar artists?
- Can we find relationships between the tags?

The Artist As A Document

- Treat the artist as a document
- Treat the tags applied to that artist as the words in the document

```
// Load the tags, and artists, get the search engine
SimpleIndexer indexer = engine.getSimpleIndexer();
for(Entry<String,List<Tag>> entry : artists.entrySet()) {
    indexer.startDocument(entry.getKey());
    for(Tag tag : entry.getValue()) {
       // Add a term with a frequency to the index
       indexer.addTerm(tag.getName(), tag.getCount());
    }
    indexer.endDocument();
}
indexer.finish();
```

Frequent Tags vs. Best Tags

- You need to be able to select the tags that best define a particular artist
  - Ones applied frequently to that artist and infrequently to others

Tags By Frequency	
classic rock	4461
rock	3281
pop	1808
british	1498
The Beatles	1147
60s	1050
oldies	939
psychedelic	847
alternative	349

Tags By Importance	
The Beatles	0.024
british invasion	0.017
60s	0.015
liverpool	0.014
british	0.014
british psychedelia	0.013
oldies	0.013
britrock	0.012
psychedelic	0.012

Computing Artist Similarity

- We can easily compute the similarity between any two artists by computing the similarity between their documents

```
// Find most similar artists by tag
DocumentVector lv =
    searchEngine.getDocumentVector(artist, "tag");
ResultSet rs = lv.findSimilar();

// Display similar artists
List<Result> results = rs.getResults(0, 10);
for (Result result : results) {
    System.out.printf("%.3f %s\n",
                      result.getScore(),
                      result.getSingleFieldValue("name"));
}
```

Web Music Explorer

DEMO

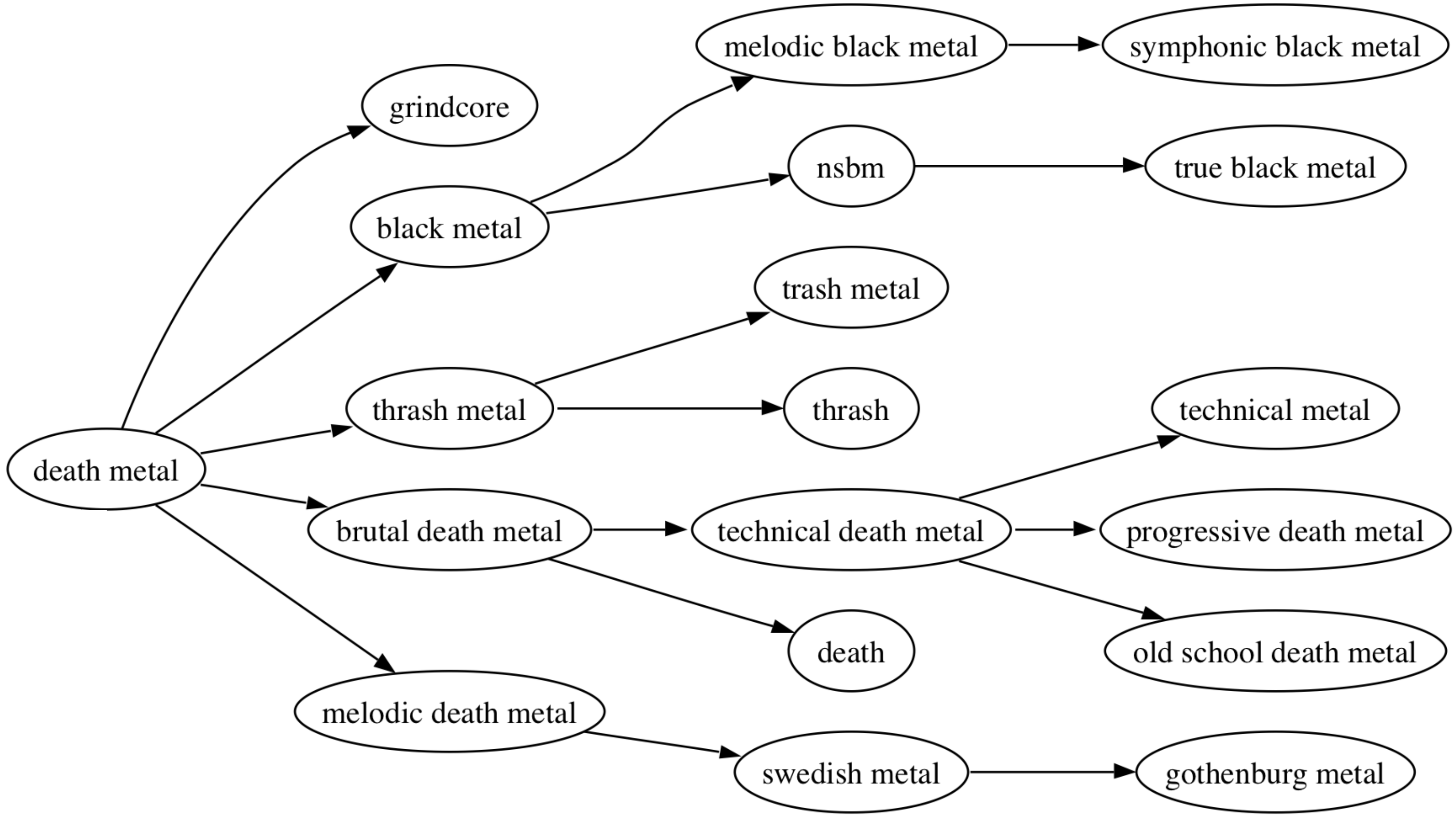
Agenda

- Background: What Is Minion?
- Minion and Lucene
- Documents and Fields
- Indexing and Querying
- Tools for the Advanced Beginner
- Tag Based Recommendation
- Tag Gardening

Building A Taxonomy Of Tags

- Treat a tag as a document
  - The words are the artists to whom the tag has been applied
- We can build a taxonomy for a tag set using the following approach
  1) Rank the tags from most to least popular
  2) Take the most popular unattached tag, *T*
  3) Attach *T* to the most popular tag whose similarity to *T* exceeds a given threshold

A Tagsonomy Of Death Metal



Tags Are Words Too!

- Synonymy
  - Multiple tags that have the same meaning
    - hip-hop, hip hop, hiphop
  - Minion can find synonyms by looking at tags that overlap
- Polysemy
  - Multiple meanings associated with a single tag
    - progressive: rock, jazz, metal,...
  - Minion can do supervised and unsupervised sense disambiguation
- These can be computed for a specific domain

For More Information

- Minion Open Source project
  - http://minion.dev.java.net
- Project Aura Session
  - TS-5841
- Steve Green's Blog
  - http://blogs.sun.com/searchguy

THANK YOU

Jeff Alexander
Stephen Green

TS-5027