Focused Crawling with

ApacheCon North America
Vancouver, 2016
Hello!

I am Sujen Shah

Computer Science @ University of Southern California

Research Intern @ NASA Jet Propulsion Laboratory

Member of The ASF and Nutch PMC since 2015

sujen@apache.org

/in/sujenshah
Outline

● The Apache Nutch Project
● Architectural Overview
● Focused Crawling
● Domain Discovery
● Evaluation
● Future Additions
● Acknowledgements
Apache Nutch

- Highly extensible and scalable open source web crawler software project.
- Hadoop based ecosystem, provides scalability.
- Highly modular architecture, to allow development of custom plugins.
- Supports full-text indexing and searching.
- Multi-threaded robust distributed crawling with configurable politeness.

Project website: http://nutch.apache.org/
Nutch History

- 2003: Started by Doug Cutting and Mike Caffarella
- 2005: MapReduce implementation and Hadoop spin off from Nutch
- 2006: Use MimeType Detection from Tika
- 2007: Top Level Project at Apache
- 2012: Nutch 2.x released offering storage abstraction via Apache Gora
- 2014: REST API, Publisher/Subscriber, JavaScript interaction and content-based Focused Crawling capabilities
Architecture

[Diagram courtesy Florian Hartl: http://florianhartl.com/nutch-how-it-works.html]
Architecture

Stores info for URLs:
- URL
- Fetch Status
- Signature
- Protocols

[Diagram courtesy Florian Hartl : http://florianhartl.com/nutch-how-it-works.html]
Architecture

Nutch

Stores incoming links to each URL and its associated anchor text.

Fetcher
Parser
Segments

Link Inverter

CrawlDB
LinkDB

Injectors

filters, normalizers, plugins (Nutch)

Indexer (Lucene)
Index

[Diagram courtesy Florian Hartl: http://florianhartl.com/nutch-how-it-works.html]
Architecture

Stores:
- Raw page content
- Parsed content, outlinks and metadata
- Fetch-list

[Diagram courtesy Florian Hartl: http://florianhartl.com/nutch-how-it-works.html]
Architecture

[Diagram courtesy Florian Hartl: http://florianhartl.com/nutch-how-it-works.html]
Nutch Workflow

Typical workflow is a sequence of batch operations

- **Inject**: Populate crawlDB from seed list
- **Generate**: Selects URLs to fetch
- **Fetch**: Fetched URLs from fetchlist
- **Parse**: Parse content from fetched URLs
- **UpdateDB**: Update the crawlDB
- **InvertLinks**: Builds the linkDB
- **Index**: Optional step to index in SOLR, Elasticsearch, etc
Few more tools at a glance

- **Fetcher**:  
  ○ Multi-threaded, high throughput  
  ○ Limit load on servers  
  ○ Partitioning by host, IP or domain

- **Plugins**:  
  ○ On demand activation  
  ○ Customizable by the developer  
  ○ Example: URL filters, protocols, parsers, indexers, scoring etc

- **WebGraph**:  
  ○ Stores outlinks, inlinks and node scores  
  ○ Iterative link analysis by LinkRank
Crawl Frontier

The crawl frontier is a system that governs the order in which URLs should be followed by the crawler.

Two important considerations [1]:

- Refresh rate: High quality pages that change frequently should be prioritized
- Politeness: Avoid repeated fetch requests to a host within a short time span

Frontier Expansion

- Manual Expansion:
  - Seeding new URLs from
    - Reference websites (Wikipedia, Alexa, etc)
    - Search engines
    - From prior knowledge

- Automatic discovery:
  - Following contextually relevant outlinks
    - Cosine similarity, Naive Bayes plugins
  - Controlling by URL filers, regular expressions
  - Using scoring
    - OPIC scoring
Broad vs. Focused Crawling

● Broad Crawling:
  ○ Unlimited crawl frontier
  ○ Limited by bandwidth and politeness factors
  ○ Useful for creating an index of the open web
  ○ Can achieve high recall
  ○ Not useful for domain discovery as crawled content may include a lot of irrelevant material

● Focused Crawling:
  ○ Limit crawl frontier by calculating relevance of URL
  ○ Low resource consumption as compared to the above
  ○ Can achieve high precision
  ○ Useful for domain discovery as it prioritizes based on content relevance
A “Domain”, here, is defined as an area of interest for a user.

Domain Discovery is the act of exploring a domain of which a user has limited prior knowledge.

Domain discovery process may include:
- Using a focused crawler
- User providing some prior knowledge in the form of text, questions or reference websites
Focused Crawling with Nutch

Previously available tools:
- URL filter plugins
  - Filter based on regular expressions
  - Whitelist/blacklist hosts
- Filter based on content mimetype
- Scoring links (OPIC scoring)
- Breadth first or Depth first crawl

Limitations:
- Follows the link structure
- Does not capture content relevance to a domain
Focused Crawling with Nutch

To capture content relevance to a domain, two new tools have been introduced.

- Cosine Similarity scoring filter
- Naive Bayes parse filter

Nutch JIRA issues:
https://issues.apache.org/jira/browse/NUTCH-2039
https://issues.apache.org/jira/browse/NUTCH-2038
Cosine Similarity

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them [1].

Similarity = \cos(\theta) = A \cdot B / |A| \cdot |B|, where A and B are the vectors.

Lesser the angle => higher the similarity

Cosine Similarity Scoring in Nutch

- Implemented as a Scoring filter
- Computed by measuring the angle between two Document Vectors.

**Document Vector**:
A term frequency vector containing all the terms occurring on a fetched page.

$$DV = \{ \text{“robots”}: 51, \text{“autonomous”}: 12, \text{“artificial”}: 23, \ldots \}$$
Cosine Similarity Scoring - Architecture

Goldstandard File

- Lowercasing
- Stopword removal
- Lucene Tokenization using Classic Tokenizer
- Porter Stemming

Creation of Document Vectors

Scores for pages:
\[
\text{similarity} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}
\]
Cosine Similarity Scoring - Working

Features of the similarity scoring plugin:
- Scores a page based on content relevance
- Leverages a simplistic bag-of-words approach
- Outlinks from relevant parent pages are considered relevant
Iteration 1

- Start with an initial seed
- Seed is considered to be relevant
- User provides keyword list for cosine similarity

Policy: Fetch top 4 urls in frontier

Unfetched (in the crawl frontier)

Fetched

Decreasing order of relevance

All children given same priority as parent in the crawl frontier
Iteration 2

- Children are fetched by the crawler
- Similarity against the goldstandard is computed and scores are assigned.

Policy: Fetch top 4 urls in frontier

- -- Unfetched (in the crawl frontier)
- Fetched
- Decreasing order of relevance
Iteration 3

Policy: Fetch top 4 urls in frontier

Unfetched (in the crawl frontier)
Fetched

Decreasing order of relevance
Iteration 4

Policy: Fetch top 4 urls in frontier

- Unfetched (in the crawl frontier)
- Fetched

Decreasing order of relevance
Iteration 5

Policy: Fetch top 4 urls in frontier

- - - - - Unfetched (in the crawl frontier)

- - - - - Fetched

- - - - - Decreasing order of relevance
Naive Bayes Classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features [1].

**Naive Bayes in Nutch**

- Implemented as a parse filter
- Classifies a fetched page relevant or irrelevant based on a user provided training dataset

Naive Bayes Classifier

Working

- User provides a set of labeled examples as training data
- Create a model based on given training data
- Classify each page as relevant (positive) or irrelevant (negative)
Naive Bayes Classifier

Working

Features:
- All outlinks from an irrelevant (negative) page are discarded
- All outlinks from a relevant (positive) page are followed
Evaluation

The following process was followed to perform domain discovery using the tools discussed earlier:

● Deploy 3 different Nutch configurations
  a. Custom Regex-filters and default scoring
  b. Cosine similarity scoring activated with keyword list
  c. Naive Bayes filter activated with labeled training data
● Provide the same seeds to all 3 configurations
● Crawl was run for 7 iterations

[Thanks to Xu Wu for the evaluations]
## Evaluation

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Regex-filters and seed list</th>
<th>Cosine similarity scoring filter</th>
<th>Naive Bayes parse filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Domain related</td>
<td>Total</td>
<td>Rate</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>47</td>
<td>36%</td>
</tr>
<tr>
<td>2</td>
<td>476</td>
<td>1286</td>
<td>37%</td>
</tr>
<tr>
<td>3</td>
<td>169</td>
<td>1334</td>
<td>13%</td>
</tr>
<tr>
<td>4</td>
<td>354</td>
<td>1351</td>
<td>26%</td>
</tr>
<tr>
<td>5</td>
<td>704</td>
<td>1553</td>
<td>45%</td>
</tr>
<tr>
<td>6</td>
<td>267</td>
<td>1587</td>
<td>17%</td>
</tr>
<tr>
<td>7</td>
<td>354</td>
<td>1715</td>
<td>21%</td>
</tr>
<tr>
<td>Total</td>
<td>2341</td>
<td>8873</td>
<td>26%</td>
</tr>
</tbody>
</table>

[Thanks to Xu Wu for the evaluations]
Evaluation

Fetch Relevance per round

% of actually relevant pages

Round number

[Thanks to Xu Wu for the evaluations]
Analysis

- Page Relevance* for the first 3 rounds is almost the same for all the methods
- Relevancy sharply rises for the Cosine similarity scoring for further rounds
- Naive Bayes and custom regex-filters perform almost the same

*Page Relevance*
“True Relevance” of a fetched page was calculated using MeaningCloud’s [1] text classification API.

Limitations

A few things to consider:

- The performance of these new focused crawling tools depends on how well the user provides the initial domain relevant data.
  - Keyword/Text for Cosine Similarity
  - Labeled text for Naive Bayes Filter
- Currently, these tools perform well with textual data, there is no provision for multimedia
- These techniques are good at providing topically relevant content, but may not provide factually relevant content
Future Improvements

Potential additions to focused crawling in Nutch:

- Use the html DOM structure of a page to assess relevance to a domain (ex- news, forums, etc)
- Augment the goldstandard in Cosine similarity with newly found highly relevant text in between iterations
- Use Tika’s NER Parser and GeoParser to extract entities and locations to capture more metadata about a domain
- Use Part-of-Speech to capture grammar(context) in a domain (ex- a same key term could occur in various domains)
Other cool tools ...

- Nutch REST API
- Publisher/Subscriber model
- Headless browsing - Selenium and PhantomJS
- Real-time graph querying of the web graph (upcoming)
Acknowledgements

Thanks to:

- Andrzej Białecki, Chris Mattmann, Doug Cutting, Julien Nioche, Mike Caffarella, Lewis John McGibbney Sebastian Nagel for ideas and material from their previous presentations
- all Nutch contributors for their amazing work!
- Florian Hartl for the architecture diagram and blogpost
- Xu Wu for the evaluations
- SlidesCarnival for the presentation template
Acknowledgements

A special thanks to:

- My mentor **Dr. Chris Mattmann** for his guidance
- The awesome team at NASA Jet Propulsion Laboratory
- And the DARPA MEMEX Program
Thanks!

Any questions?

You can find me at:
@sujen
sujen@apache.org