ClusteringPlugin

Clustering Plugin

plugin name

${renderedContent}

plugin version

${renderedContent}

Plugin Info

- provider: The Carrot2 project, http://www.carrot2.org
- plugin home url: Plugin is included in Nutch codebase.
- plugin download url: Binaries included with Nutch.
- license: BSD-style
- short description: Plugin for clustering search results at query-time.
- long description: This plugin organizes search results into groups of (related, hopefully) documents.
- configureable parameters: Take a look at the defaults defined in nutch-default.xml (search for 'clustering').
- meta data added to index: None. Clustering is performed dynamically for each result set.
- required jars: The entire `lib` folder in the plugin must be present in classpath. More JARs might be needed from the Carrot2 project if additional algorithms or languages are to be used.
- plugin extension point interface: `net.nutch.clustering.OnlineClusterer`
- **Carrot2 JARs come from codebase in version: 2.1**

Installation guide

- Create a search index using the instructions provided in Nutch documentation.
- Deploy Nutch Web application and make sure the index is found and searching works (type a query and see if you get any results).
- Stop the web server (Tomcat, Jetty or anything you like).
- Modify `WEB-INF/classes/nutch-default.xml` file and include the clustering plugin (it is by default ignored) by adding `clustering-carrot2` to `plugin.includes` property.
- Restart your web server and reload the search page. You should see the `clustering` checkbox next to `search` button. Enable it and rerun your query. Cluster labels and documents should appear to the right of search results.

Note that the user interface in default Nutch's Web application is very limited and you'll most likely need something more application-specific. Look at http://www.carrot2.org or http://www.carrot-search.com for inspiration.

Configuration guide

Libraries in this release are precompiled with stemming and stop words for various languages present in the Carrot2 codebase. You should define the default language and supported languages in Nutch configuration file (nutch-site.xml). If nothing is given in Nutch configuration English is used by default. The following properties can be added to `nutch-site.xml`:

```
<!-- Carrot2 Clustering plugin configuration -->

<property>
  <name>extension.clustering.carrot2.defaultLanguage</name>
  <value>en</value>
  <description>Two-letter ISO code of the language.
  http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt</description>
</property>

<property>
  <name>extension.clustering.carrot2.languages</name>
  <value>en,nl,da,fi,fr,de,it,no,pl,pt,ru,es,sv,tr,ro,hu</value>
  <description>All languages to be used by the clustering plugin.
  This list includes all currently supported languages (although not all of them
  will successfully instantiate -- support for Polish requires additional
  libraries for instance). Adjust to your needs, fewer languages take less
  memory.

  If you use the language recognizer plugin, then each hit will come with its
  own ISO language code. All hits with no explicit language take the default
  language specified in "extension.clustering.carrot2.defaultLanguage" property.
  </description>
</property>
```

Using other Carrot2 clustering algorithms

To limit the size of the clustering plugin, the default implementation is shipped with the Lingo algorithm – just one of several alternatives available in the Carrot2 project. This section describes how to substitute the default algorithm with a different one.

First, prepare the following:

- Install Nutch, enable the clustering plugin and make sure it works in the default configuration.
- Get a precompiled distribution of Carrot2 (http://project.carrot2.org/download.html), for example the DCS demo, or compile it from scratch.

Now you are ready to install a different clustering algorithm. The instructions below show how to run STC (Suffix Tree Clustering) instead of Lingo on the Jetty server (6.1.5). We will use a binary release of the DCS as a source of the required Carrot2 JARs.

- Make a symbolic link from `webapps/ROOT.war` to a compile Nutch WAR file (or just place it there).
- Make sure clustering and search work as expected.
- Download the binary release of DCS (we will use the one from the latest stable release: version 2.1).
- Carrot2 framework has the notion of a "process" – a pipeline of components that process search results and emit clusters. We will need to provide the name of an XML file which defines such a process to Nutch's clustering extension and give it access to all the required classes it may need. Let's start from defining a process. Unpack the DCS distribution and locate the `descriptors` folder. You'll see a bunch of files inside, the one that interests us is called `alg-stc-en.xml`. Its contents should look like this:

```
<local-process id="stc-en">
  <name>STC (+English)</name>
  <description>Suffix Tree Clustering Algorithm</description>

  <input  component-key="input-demo-webapp" />

  <filter component-key="filter-language-detection-en" />
  <filter component-key="filter-tokenizer" />
  <filter component-key="filter-case-normalizer" />
  <filter component-key="filter-stc" />

  <output component-key="output-demo-webapp" />
</local-process>
```

- Now edit the above file and change `input` component key to `nutch-input` and `output` component key to `output-array`, leaving everything else exactly as it were.

```
<local-process id="stc-en">
  <name>STC (+English)</name>
  <description>Suffix Tree Clustering Algorithm</description>

  <input  component-key="input-nutch" />

  <filter component-key="filter-language-detection-en" />
  <filter component-key="filter-tokenizer" />
  <filter component-key="filter-case-normalizer" />
  <filter component-key="filter-stc" />

  <output component-key="output-array" />
</local-process>
```

- The filters you see in the process descriptor should also be available. Some of them are built in in the Carrot2 core, other should be copied from DCS's distribution to the same temporary folder we copied the process definition to. In our case the following filter definition files should be copied: `filter-language-detection-en.bsh`, `filter-tokenizer.bsh`, `filter-case-normalizer.bsh` and `filter-stc.bsh`.
- Process and component descriptors are read as a resource (relative to classpath). Jetty can be configured to have an additional classpath entry as a folder, but it slightly complicates things (hierarchy of classloaders may result in some hard-to-track errors). It will be easier to just place all the required stuff in Nutch's Web application context under `WEB-INF`. If you work with WAR file directly, you'll need to add the resources mentioned below to the WAR file (it's a ZIP file).
- Copy process and component descriptor files to `{NUTCH-CONTEXT}/WEB-INF/classes/`.
- Copy certain JAR files from the DCS (`WEB-INF/lib/*.jar`) to `{NUTCH-CONTEXT}/WEB-INF/lib`. Which JARs should be copied is not an easy question to answer. In general, you can copy everything that won't clash with your Web container. We suggest *not* to copy the following: `carrot2-util-log4j*.jar` (log4j configuration files), `commons-logging*.jar` (clashes with Nutch's version), `jasper*.jar` and `org.mortbay*.jar` (already present in Web containers). The rest should be safe to just copy, overwriting anything present in Nutch.
- Finally, the path to the clustering process should be added to `{NUTCH-CONTEXT}/WEB-INF/classes/nutch-site.xml`:

```
<property>
  <name>extension.clustering.carrot2.process-resource</name>
  <value>/alg-stc-en.xml</value>
</property>
```

- Restart your Web application container. The clustering plugin should use STC clustering algorithm if everything was ok. If something is wrong, inspect your log files – they usually indicate the problem (i.e., missing classes) quite clearly.