LanguageDetection

Solr's Language Detection

⚠ Solr3.5

Introduction

This feature adds the ability to detect the language of a document before indexing and then make appropriate decisions about analysis, etc. It is implemented as an UpdateRequestProcessor, and there are two implementations:

- Tika implementation based upon Tika's language detection capabilities, which covers many, but not all, languages. See http://tika.apache.org/0.10/detection.html for more information on the languages supported.
- LangDetect implementation based upon http://code.google.com/p/language-detection/ which supports more languages (53) and has some advanced CJK support.

The component also supports automatic renaming of fields according to detected language and other advanced parameters, all explained in the next section.

Configuration

The UpdateRequestProcessor is configured in solrconfig.xml, and supports many parameters. All parameters listed may also be overridded on the update request itself. A minimal configuration specifies the input fields for language identification as well as the output field for the detected language code:

```
<processor class="org.apache.solr.update.processor.TikaLanguageIdentifierUpdateProcessorFactory">
   <lst name="defaults">
     <str name="langid.fl">title,subject,text,keywords</str>
     <str name="langid.langField">language_s</str>
   </lst>
</processor>
```

Alternatively, using the implementation based on http://code.google.com/p/language-detection/

```
<processor class="org.apache.solr.update.processor.LangDetectLanguageIdentifierUpdateProcessorFactory">
   <lst name="defaults">
     <str name="langid.fl">title,subject,text,keywords</str>
     <str name="langid.langField">language_s</str>
   </lst>
</processor>
```

NOTE: The processor supports the `defaults/appends/invariants` concept for its config. However, it is also possible to skip this level and configure the parameters directly underneath the `<processor>` tag.

Below follows a list of each configuration parameters and their meaning:

langid

Lets you enable/disable this processor

Value: true/false

Default: true

langid.fl

Specifies the list of field names to take as input for the language detection

Value: Same format as `fl`, i.e. a comma or space delimited list of field names

Default: N/A (This parameter is mandatory)

langid.langField

Specifies the field to output detected language into. The value written is the language code as emitted by Tika or `LangDetect`.

Value: Name of field

Default: N/A (This parameter is mandatory)

langid.langsField

Specifies the field to output a list of detected languages into. This must be a multiValued String field. If you use `langid.map.individual`, each detected language will be added to this field.

Value: Name of field

Default: (Empty - Nothing is written by default)

langid.overwrite

Specifies whether the output in `langField` and `langsField` shall be overwritten if `langField` already contains a value. If not set and `langField` contains a value, `langField` will be subject to white list filtering and then copied to `langsField`, which will be overwritten.

Value: true/false

Default: false

langid.threshold

Specifies a threshold between 0-1 for how close the language identification match must be before being accepted. For long texts a high value like 0.8 will give the best results, but for shorter texts you may need to specify lower thresholds, and at the same time risking a lower quality detection. Experiment on your data to find a good value.

Value: A float value between 0.0 and 1.0

Default: 0.5

langid.whitelist

Specifies an optional list of language codes that shall be the only allowed outputs from language identification. This means that if another language is detected, it will not be accepted and you'll fall back to fallback language. This is great in combination with langid.map=true to make sure you only index documents into fields that exist in your schema.

Value: A comma separated list of language codes accepted. Note that these are codes as output from your detector **before** mapping with langid.map.lcmap

Default: (Empty - all languages are allowed)

langid.map

To enable field name mapping, set langid.map=true. It will then map field names for all fields in langid.fl.

If the set of fields to map is different from langid.fl, supply langid.map.fl. Those fields will then be renamed with a language suffix equal to the language detected from the langid.fl fields.

Value: true/false

Default: false

langid.map.fl

Optional list of fields to do field name mapping for. See langid.map

Value: A comma separated list of fields

Default: (Empty - by default all fields in langid.fl will be mapped)

langid.map.keepOrig

If set to true, the mapping operation will leave the original field in place, i.e. it will act as a field copy instead of a move/map.

Value: true/false

Default: false

langid.map.individual

If you require detecting languages separately for each field, supply langid.map.individual=true. The supplied fields will then be renamed according to detected language on an individual field basis.

Value: true/false

Default: false

langid.map.individual.fl

If the set of fields to detect individually is different from the already supplied langid.fl or langid.map.fl, supply langid.map.individual.fl. The fields listed in langid.map.individual.fl will then be detected individually, while the rest of the mapping fields will be mapped according to global document language.

Value: A comma separated list of fields

Default: (Empty - by default all fields in langid.fl or langid.map.fl will be mapped)

langid.fallbackFields

If no language is detected with sufficient score (see langid.threshold), or if the detected language is not in the whitelist (see langid.whitelist), we will lookup the field(s) from `langid.fallbackFields` one by one to see if we find a language code. If found it will be used as the fallback language. If not, we will continue to look for `langid.fallback`

Value: Comma separated list of field names in which to look for language code. May be only one as well.

Default: (Empty - not used)

langid.fallback

If no language is detected with sufficient score (see langid.threshold), or if the detected language is not in the whitelist (see langid.whitelist), and no value is found in your fallbackField, the language code specified in langid.fallback will be used. Note that if neither `fallbackFields` nor `fallback` is specified, and a language cannot be detected, language will be set to an empty string, causing potential problems further down the chain.

Value: Language code to use as fallback

Default: (Empty - not used)

langid.map.lcmap

If this parameter is specified, it will be used as a language code map. A typical usage is to map multiple detected languages to the same field name. I.e. to map both Japanese, Korean and Chinese texts to the same schema field "*_cjk", do: `langid.map.lcmap=ja:cjk zh:cjk ko:cjk`. Another use is if your language identification outputs something like en_US or en_GB but you want only one field with *_en, you say `langid.map.lcmap=en_GB:en en_US:en`. Note that this setting does not affect the language codes written to langField.

Value: A space separated list of language code mappings, on the form <from>:<to>

Default: (Empty - not used)

langid.lcmap

If this parameter is specified, it will map the language code which is output to `langField`, and also affect the field name ending. In the last example used for `langid.map.lcmap`, both `en_GB` and `en_US` would result in field name `text_en` being used, but the `language` field on the document would still contain the two variants. By instead using `langid.map=en_GB:en en_US:en` the output in the `langField` would also be normalized to simply `en`. I.e. this option performs the mapping both for language field output and for field name mapping. If you need a different field name mapping, then specify both `langid.lcmap` and `langid.map.lcmap` together.

Value: A space separated list of language code mappings, on the form <from>:<to>

Default: (Empty - not used)

langid.map.pattern and langid.map.replace

Default field mapping is <field>_<lang>, however you can define your own mapping pattern using `langid.map.pattern` and `langid.map.replace`. You may use normal Java regEx matching with groups. The text "{lang}" in the pattern will be replaced with the detected language code (or the mapped equivalent).

Value: `pattern` is a java style regex pattern and `replace` is a java style replace

Default: (Empty - not used)

langid.enforceSchema

Normally the processor will throw an exception if the result of a mapping is not a valid schema field. By enabling this option, you turn off validation of field names against schema. This can be useful if you want to rename or delete fields later in the UpdateChain, i.e. you know what you're doing.

Value: true/false

Default: true

Examples

Detect and map Scandinavian languages with Tika and fallback to generic for other languages

```
<processor class="org.apache.solr.update.processor.TikaLanguageIdentifierUpdateProcessorFactory">
  <bool name="langid">true</bool>
  <str name="langid.fl">title,body</str>
  <str name="langid.langField">language</str>
  <str name="langid.whitelist">no,sv,da</str>
  <bool name="langid.map">true</bool>
  <str name="langid.fallback">generic</str>
</processor>
```

Caveats

Since the implementations uses an n-gram based approach to detection, they are susceptible to poor detection on especially short inputs. The threshold you specify in langid.threshold is normalized to match a certain similarity score in Tika, but this is not reliable for thresholds lower than 0.8. In the future, the detection quality may be improved due to changes in Tika or use of other language detection libraries.

Resources

- Apache Tika
- Language detection Library for Java
- SOLR-1979