TextProfileSignature

TextProfileSignature calculates a fuzzy hash of textual fields for Deduplication, and may be incorporated using a SignatureUpdateProcessorFactory definition including the following parameters:

Name	Type	Description	Default value
`minTokenLen`	int	The minimum token length to consider	2
`quantRate`	float	When multiplied by the maximum token frequency, this determines count quantization	.01

The signature calculation proceeds as follows:

Tokenization and normalization

- Tokens are contiguous alphanumeric characters
- Normalized to lowercase
- Discarded if shorter than `minTokenLen`

Tokens are then counted, tracking the frequency `maxFreq` of the most frequent token.

Count quantization

A value `quant` is calculated as follows:

	1	if `maxFreq` <= 1
`quant :=`	2	if round(`maxFreq * quantRate`) < 2
	round(`maxFreq * quantRate`)	otherwise

Token frequencies are then rounded down to the nearest multiple of `quant`, and any token occurring less than `quant` times is discarded.

Hashing

The set of frequencies is transformed to a string as a space-delimited sequence of tokens and their frequencies, in descending frequency order. This is then MD5-hashed.

See also TextProfileSignature's javadoc

Implications and limitations

Though this matches two texts approximately, it is still based on exactly matching a single hash. It may fail to match documents that differ by exactly one word, if that word's frequency changes from `k * quant - 1` to `k * quant`.

Words appearing once are ignored unless the text consists only of words appearing once. Hence, "the cat sat on a mat" will hash distinctly to "the cat sat on the mat".

For the default `quantRate` (0.01), quant will exceed 2 only if the most frequent word occurs `maxFreq >= 251` times.

These properties all suggest that TextProfileSignature is brittle for short texts.

TextProfileSignature operates on raw text, without the filtering provided by Analyzers, and hence will fail to ignore HTML, normalize for diacritics, word stem /semantics, or incorporate the relative importance of different tokens, etc. It also considers only the bag of words, ignoring any word order.

Configuration

solrconfig.xml

Example settings:

```
<!-- An example dedup update processor that creates the "id" field on the fly
     based on the hash code of some other fields.  This example has overwriteDupes
     set to false since we are using the id field as the signatureField and Solr
     will maintain uniqueness based on that anyway. -->
<updateRequestProcessorChain name="dedupe">
  <processor class="org.apache.solr.update.processor.SignatureUpdateProcessorFactory">
    <bool name="enabled">true</bool>
    <bool name="overwriteDupes">false</bool>
    <str name="signatureField">id</str>
    <str name="fields">name,features,cat</str>
    <str name="signatureClass">org.apache.solr.update.processor.TextProfileSignature</str>
    <str name="quantRate">.2</str>
  </processor>
  <processor class="solr.LogUpdateProcessorFactory" />
  <processor class="solr.RunUpdateProcessorFactory" />
</updateRequestProcessorChain>
```