

LanguageAnalysis

Language Analysis

This page describes some of the language-specific analysis components available in Solr. These components can be used to improve search results for specific languages.

Please look at [AnalyzersTokenizersTokenFilters](#) for other analysis components you can use in combination with these components.

NOTE: This page is mostly **obsolete**. The [Solr Example](#) now contains configurations for various languages as fieldTypes (text_XX). This is synchronized with the support from Lucene.

- [Language Analysis](#)
 - [By language](#)
 - [Arabic](#)
 - [Armenian](#)
 - [Basque](#)
 - [Brazilian Portuguese](#)
 - [Bulgarian](#)
 - [Catalan](#)
 - [Chinese, Japanese, Korean](#)
 - [Czech](#)
 - [Danish](#)
 - [Dutch](#)
 - [English](#)
 - [Finnish](#)
 - [French](#)
 - [Galician](#)
 - [German](#)
 - [Greek](#)
 - [Hebrew](#)
 - [Hindi](#)
 - [Hungarian](#)
 - [Indonesian](#)
 - [Italian](#)
 - [Lao, Myanmar, Khmer](#)
 - [Norwegian](#)
 - [Persian / Farsi](#)
 - [Polish](#)
 - [Portuguese](#)
 - [Romanian](#)
 - [Russian](#)
 - [Spanish](#)
 - [Swedish](#)
 - [Thai](#)
 - [Turkish](#)
 - [My language is not listed!!!](#)
 - [Other Tips](#)
 - [Tokenization](#)
 - [Ignoring Case](#)
 - [Ignoring Diacritics](#)
 - [Stopwords](#)
 - [Stemming](#)
 - [Notes about solr.HunspellStemFilterFactory](#)
 - [Notes about solr.PorterStemFilterFactory](#)
 - [Notes about solr.KStemFilterFactory](#)
 - [Notes about solr.SnowballPorterFilterFactory](#)
 - [Customizing Stemming](#)
 - [solr.KeywordMarkerFilterFactory](#)
 - [solr.StemmerOverrideFilterFactory](#)
 - [Decompounding](#)

By language

Arabic

Solr provides support for the [Light-10](#) stemming algorithm, and Lucene includes an example stopwords list.

This algorithm defines both character normalization and stemming, so these are split into two filters to provide more flexibility.

```
...
<filter class="solr.ArabicNormalizationFilterFactory"/>
<filter class="solr.ArabicStemFilterFactory"/>
...
```

Example set of Arabic [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Armenian

⚠ Solr3.1

Solr includes support for stemming Armenian via `solr.SnowballPorterFilterFactory`, and Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Armenian" />
...
```

Example set of Armenian [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Basque

⚠ Solr3.1

Solr includes support for stemming Basque via `solr.SnowballPorterFilterFactory`, and Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Basque" />
...
```

Example set of Basque [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Brazilian Portuguese

Solr includes a modified version of the Snowball Portuguese algorithm for Brazilian Portuguese, and Lucene includes an example stopwords list. This stemmer handles diacritical marks differently than the European Portuguese stemmer.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.BrazilianStemFilterFactory"/>
...
```

Example set of Brazilian Portuguese [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Bulgarian

⚠ Solr3.1

Solr includes a light stemmer for Bulgarian, following this [algorithm](#), and Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.BulgarianStemFilterFactory"/>
...
```

Example set of Bulgarian [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Catalan

⚠ Solr3.1

Solr includes support for stemming Catalan via `solr.SnowballPorterFilterFactory`, and Lucene includes an example stopwords list.


```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Catalan" />
...
```

Example set of Catalan [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Chinese, Japanese, Korean

Lucene provides support for these languages with CJKTokenizer, which indexes bigrams and does some character folding of full-width forms.

```
<tokenizer class="solr.CJKTokenizerFactory"/>
...
```


 [Solr3.1](#) Alternatively, for Simplified Chinese, Solr provides support for Chinese word segmentation `solr.SmartChineseWordTokenFilterFactory` in the analysis-extras contrib module. This component includes a large dictionary and segments Chinese text into words with the Hidden Markov Model. To use this filter, see `solr/contrib/analysis-extras/README.txt` for instructions on which jars you need to add to your SOLR_HOME/lib

To use the default setup with fallback to English Porter stemmer for english words, use:

```
<analyzer class="org.apache.lucene.analysis.cn.smart.SmartChineseAnalyzer"/>
```

Or to configure your own analysis setup, use the [SmartChineseSentenceTokenizerFactory](#) along with your custom filter setup. The sentence tokenizer tokenizes on sentence boundaries and the [SmartChineseWordTokenFilter](#) breaks this further up into words.

```
<analyzer>
  <tokenizer class="solr.SmartChineseSentenceTokenizerFactory"/>
  <filter class="solr.SmartChineseWordTokenFilterFactory"/>
  <filter class="solr.LowerCaseFilterFactory"/>
  <filter class="solr.PositionFilterFactory" />
</analyzer>
```

 Note: Be sure to use [PositionFilter](#) at query-time (only) as these languages do not use spaces between words.

Czech

 [Solr3.1](#)

Solr includes a light stemmer for Czech, following this [algorithm](#), and Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.CzechStemFilterFactory"/>
...
```

Example set of Czech [stopwords](#) (Be sure to switch your browser encoding to UTF-8))

Danish

Solr includes support for stemming Danish via `solr.SnowballPorterFilterFactory`, and Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Danish" />
...
```

Example set of Danish [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

 Note: See also [Decompounding](#) below.

Dutch

Solr includes two stemmers for Dutch via `solr.SnowballPorterFilterFactory`, and Lucene includes an example stopword list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Dutch" />
...
```

An alternative stemmer (Kraaij-Pohlmann) can be used by specifying the language as "Kp".

Example set of Dutch [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: See also [Decompounding](#) below.

English

Solr includes three stemmers for English: the original Porter stemmer via `solr.PorterStemFilterFactory`, the Porter2 stemmer via `solr.SnowballPorterFilterFactory`, and a plural-only stemmer ⚠ [Solr3.1](#) via `solr.EnglishMinimalStemFilterFactory`. Lucene includes an example stopword list from the snowball project.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.PorterStemFilterFactory"/>
...
```

⚠ Note: The standard `PorterStemFilterFactory` is significantly faster than `solr.SnowballPorterFilterFactory`.

Larger example set English [stopwords](#)

Finnish

Solr includes two stemmers for Finnish: one via `solr.SnowballPorterFilterFactory`, and an alternative stemmer ⚠ [Solr3.1](#) via `solr.FinnishLightStemFilterFactory`. Lucene includes an example stopword list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Finnish" />
...
```

Example set of Finnish [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: See also [Decompounding](#) below.

⚠ Note: The Snowball stemmer for Finnish has known bugs, due to a bug in [snowball itself](#)

French

Solr includes three stemmers for French: one via `solr.SnowballPorterFilterFactory`, an alternative stemmer ⚠ [Solr3.1](#) via `solr.FrenchLightStemFilterFactory`, and an even less aggressive approach ⚠ [Solr3.1](#) via `solr.FrenchMinimalStemFilterFactory`. Solr can also removing elisions via `solr.ElisionFilterFactory`, and Lucene includes an example stopword list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.ElisionFilterFactory"/>
<!-- do word delimiter, etc here -->
<filter class="solr.SnowballPorterFilterFactory" language="French" />
...
```

Example set of French [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: Its probably best to use the [ElisionFilter](#) before [WordDelimiterFilter](#). This will prevent very slow phrase queries.

Galician

⚠ [Solr3.1](#)

Solr includes a stemmer for Galician following this [algorithm](#), and Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.GalicianStemFilterFactory"/>
...
```

Example set of Galician [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

German

Solr includes support for stemming German with five different algorithms: two via `solr.SnowballPorterFilterFactory`, one via `solr.GermanStemFilterFactory`, a lightweight stemmer ⚠ [Solr3.1](#) via `solr.GermanLightStemFilterFactory`, and an even less aggressive approach ⚠ [Solr3.1](#) via `solr.GermanMinimalStemFilterFactory`. Lucene includes an example stopwords list.

With the `solr.SnowballPorterFilterFactory` you can supply two different language attributes: "German" and "German2". German2 is just a modified version of German that handles the umlaut characters differently: for example it treats "ü" as "ue" in most contexts.

The `solr.GermanStemFilterFactory` instead uses a different [algorithm](#).

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="German2" />
...
```

Example set of German [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: See also [Decompounding](#) below.

Greek

Solr includes support for stemming Greek following this [algorithm](#) ⚠ [Solr3.1](#), as well as support for case/diacritics-insensitive search via `solr.GreekLowerCaseFilterFactory`, and Lucene includes an example stopwords list.

```
...
<filter class="solr.GreekLowerCaseFilterFactory"/>
<filter class="solr.GreekStemFilterFactory"/>
...
```

Example set of Greek [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: Be sure to use the Greek-specific [GreekLowerCaseFilterFactory](#)

Hebrew

```
...
<tokenizer class="solr.ICUTokenizerFactory"/>
...
```

Example set of Hebrew [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Hindi


⚠ [Solr3.1](#)

Solr includes support for stemming Hindi following this [algorithm](#), support for common spelling differences via `solr.HindiNormalizationFilterFactory` following this [algorithm](#), support for encoding differences via `solr.IndicNormalizationFilterFactory` following this [algorithm](#), and Lucene includes an example stopwords list.

```
...
<filter class="solr.IndicNormalizationFilterFactory"/>
<filter class="solr.HindiNormalizationFilterFactory"/>
<filter class="solr.HindiStemFilterFactory"/>
...
```


Example set of Hindi [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Hungarian

Solr includes two stemmers for Hungarian: one via `solr.SnowballPorterFilterFactory`, and an alternative stemmer  [Solr3.1](#) via `solr.HungarianLightStemFilterFactory`. Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Hungarian" />
...
```

Example set of Hungarian [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

 Note: See also [Decompounding](#) below.

Indonesian

 [Solr3.1](#)


Solr includes support for stemming Indonesian (Bahasa Indonesia) following this [algorithm](#), and Lucene includes an example stopwords list.

You can set the `stemDerivational` attribute to false to only stem inflectional suffixes, for a lighter approach.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.IndonesianStemFilterFactory" stemDerivational="true" />
...
```

Example set of Indonesian [stopwords](#)

Italian

Solr includes two stemmers for Italian: one via `solr.SnowballPorterFilterFactory`, and an alternative stemmer  [Solr3.1](#) via `solr.ItalianLightStemFilterFactory`. Lucene includes an example stopwords list.


```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Italian" />
...
```

Example set of Italian [stopwords](#) (Be sure to switch your browser encoding to UTF-8)


Lao, Myanmar, Khmer

 [Solr3.1](#)

Lucene provides support for segmenting these languages into syllables with `solr.ICUTokenizerFactory` in the analysis-extras contrib module. To use this tokenizer, see `solr/contrib/analysis-extras/README.txt` for instructions on which jars you need to add to your `SOLR_HOME/lib`

 Note: Be sure to use [PositionFilter](#) at query-time (only) as these languages do not use spaces between words.

Norwegian

Solr includes support for stemming Norwegian via `solr.SnowballPorterFilterFactory`, and Lucene includes an example stopwords list. Since  [Solr3.6](#) you can also use `solr.NorwegianLightStemFilterFactory` for a lighter variant or `solr.NorwegianMinimalStemFilterFactory` attempting to normalize plural endings only. These two are simple rule based stemmers, not handling exceptions or irregular forms.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Norwegian" />
...
```

Example set of Norwegian [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: See also [Decompounding](#) below.

Persian / Farsi

Solr includes support for normalizing Persian via `solr.PersianNormalizationFilterFactory`, and Lucene includes an example stopwords list.

```
...
<filter class="solr.ArabicNormalizationFilterFactory"/>
<filter class="solr.PersianNormalizationFilterFactory"/>
...
```

Example set of Persian [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: [WordDelimiterFilter](#) does not split on joiners by default. You can solve this by using [ArabicLetterTokenizerFactory](#), which does, or by using a custom [WordDelimiterFilterFactory](#) which supplies a customized `charTypeTable` to [WordDelimiterFilter](#). In either case, consider using [PositionFilter](#) at query-time (only), as the [QueryParser](#) does not consider joiners and could create unwanted phrase queries.

Polish

⚠ [Solr3.1](#)

Lucene provides support for Polish stemming `solr.StempelPolishStemFilterFactory` in the `analysis-extras` contrib module. This component includes an algorithmic stemmer with tables for Polish. To use this filter, see `solr/contrib/analysis-extras/README.txt` for instructions on which jars you need to add to your `SOLR_HOME/lib`

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.solr.StempelPolishStemFilterFactory"/>
...
```

Example set of Polish [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Portuguese

Solr includes four stemmers for Portuguese: one via `solr.SnowballPorterFilterFactory`, an alternative stemmer ⚠ [Solr3.1](#) via `solr.PortugueseStemFilterFactory`, a lighter stemmer ⚠ [Solr3.1](#) via `solr.PortugueseLightStemFilterFactory`, and an even less aggressive approach ⚠ [Solr3.1](#) via `solr.PortugueseMinimalStemFilterFactory`. Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Portuguese" />
...
```

Example set of Portuguese [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Romanian

Solr includes support for stemming Romanian via `solr.SnowballPorterFilterFactory`, and Lucene includes an example stopwords list.

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Romanian" />
...
```

Example set of Romanian [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Russian

Solr includes two stemmers for Russian: one via `solr.SnowballPorterFilterFactory`, and an alternative stemmer ⚠ [Solr3.1](#) via `solr.RussianLightStemFilterFactory`. Lucene includes an [example stopword list](#).

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Russian" />
...
```

Example set of Russian [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Spanish

Solr includes two stemmers for Spanish: one via `solr.SnowballPorterFilterFactory`, and an alternative stemmer ⚠ [Solr3.1](#) via `solr.SpanishLightStemFilterFactory`. Lucene includes an [example stopword list](#).

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Spanish" />
...
```

Example set of Spanish [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

Swedish

Solr includes two stemmers for Swedish: one via `solr.SnowballPorterFilterFactory`, and an alternative stemmer ⚠ [Solr3.1](#) via `solr.SwedishLightStemFilterFactory`. Lucene includes an [example stopword list](#).

```
...
<filter class="solr.LowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Swedish" />
...
```

Example set of Swedish [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: See also [Decompounding](#) below.

Thai

Solr includes support for breaking Thai text into words via `solr.ThaiWordFilterFactory`

```
...
<filter class="solr.ThaiWordFilterFactory"/>
...
```

⚠ Note: Be sure to use [PositionFilter](#) at query-time (only) as this language does not use spaces between words.

Turkish

Solr includes support for stemming Turkish via `solr.SnowballPorterFilterFactory`, as well as support for case-insensitive search via `solr.TurkishLowerCaseFilterFactory` ⚠ [Solr3.1](#), and Lucene includes an [example stopword list](#). ⚠ [Solr4.8](#) includes `solr.ApostropheFilterFactory` for apostrophe handling suitable for Turkish.

```
...
<filter class="solr.ApostropheFilterFactory"/>
<filter class="solr.TurkishLowerCaseFilterFactory"/>
<filter class="solr.SnowballPorterFilterFactory" language="Turkish" />
...
```

Example set of Turkish [stopwords](#) (Be sure to switch your browser encoding to UTF-8)

⚠ Note: Be sure to use the Turkish-specific [TurkishLowerCaseFilterFactory](#) ⚠ [Solr3.1](#)

My language is not listed!!!

Your language might work anyway. A first step is to start with the "textgen" type in the example schema. Remember, things like stemming and stopwords aren't mandatory for the search to work, only optional language-specific improvements.

If you have problems (your language is highly-inflectional, etc), you might want to try using an n-gram approach as an alternative.

Other Tips

Tokenization

In general most languages don't require special tokenization (and will work just fine with Whitespace + [WordDelimiterFilter](#)), so you can safely tailor the English "text" example schema definition to fit.

Ignoring Case

In most cases [LowerCaseFilterFactory](#) is sufficient. However, some languages have special casing properties, and these have their own filters:

- [TurkishLowerCaseFilterFactory](#): Use this instead of [LowerCaseFilterFactory](#) for the Turkish language. It includes special handling for [dotted and dotless İ](#).
- [GreekLowerCaseFilterFactory](#): Use this instead of [LowerCaseFilterFactory](#) for the Greek language. It removes Greek diacritics and has special handling for the Greek final sigma.

Ignoring Diacritics

Some languages use diacritics, but people are not always consistent about typing them in queries or documents.

If you are using a stemmer, most stemmers (especially Snowball stemmers) are a bit forgiving about diacritics, and these are handled on a language-specific basis.

For Latin-script writing systems, you can remove all diacritics with [ASCIIFoldingFilterFactory](#). But this might not be the best approach for your language, for example you may want ü to match to ue for German. In this case it is better to not use [ASCIIFoldingFilter](#) before stemming, but instead to use the "German2" stemmer first, which has language-specific handling for this case.

For some languages in non-Latin writing systems (Arabic, Greek, Hindi, Persian), there are filters to support the idea of "diacritics-insensitive search" already included in Solr. These filters are described above under the relevant languages.

For other languages, the [ASCIIFoldingFilterFactory](#) won't do the foldings that you need. One solution is to use `solr.analysis.ICUFoldingFilterFactory` ⚠ [similar idea](#)], which implements a [<http://unicode.org/reports/tr30/tr30-4.html> across all of Unicode

Stopwords

Stopwords affect Solr in three ways: relevance, performance, and resource utilization.

From a relevance perspective, these extremely high-frequency terms tend to throw off the scoring algorithm, and you won't get very good results if you leave them. At the same time, if you remove them, you can return bad results when the stopword is actually important.

From a performance perspective, if you keep stopwords, some queries (especially phrase queries) can be very slow.

From a resource utilization perspective, if you keep stopwords, the index is much larger than if you remove them.

One tradeoff you can make if you have the disk space: You can use [CommonGramsFilter](#) [CommonGramsQueryFilter](#) instead of [StopFilter](#). This solves the relevance and performance problems, at the expense of even more resource utilization, because it will form bigrams of stopwords to their adjacent words.

Stemming

Stemming can help improve relevance, but it can also hurt.

There is no general rule for whether or not to stem: It depends not only on the language, but also on the properties of your documents and queries.

Lucene/Solr provides different stemmers, and for some languages you may have multiple choices. Some are algorithmic based, others are dictionary based.

The Snowball stemmers rely on algorithms and considered fairly aggressive, but for many languages (see above) Solr provides alternatives that are less aggressive. In many situations a lighter approach yields better relevance: often "less is more". The light stemmers typically target the most common noun/adjective inflections, and perhaps a few derivational suffixes. The minimal stemmers are even more conservative and may only remove plural endings. The new Hunspell stemmers are both dictionary and rule based and may provide a tighter stemming than Snowball for some languages.

In general, if the language is highly inflectional, its worth evaluating at least a light stemmer as it might bring a significant improvement. Or you may consider [Hunspell](#) which have advanced rules combined with dictionaries of legal stems. Some annoyances caused by stemming can then be handled with tuning: See [Customizing Stemming](#) below.

⚠ NOTE: If stemming does not give enough precision for your requirements you may consider [lemmatization](#). No lemmatizers are included with Solr, but there exist lemmatizers both commercial and open source.

Notes about solr.HunspellStemFilterFactory

⚠ Solr3.5 The Hunspell stemmers are configured through the [HunspellStemFilterFactory](#) combined with a dictionary and an affix file. Hunspell supports 99 languages.

Notes about solr.PorterStemFilterFactory

Porter stemmer for the English language.

Standard Lucene implementation of the [Porter Stemming Algorithm](#), a normalization process that removes common endings from words.

- Example: "riding", "rides", "horses" ==> "ride", "ride", "hors".

Note: This differs very slightly from the "Porter" algorithm available in `solr.SnowballPorterFilter`, in that it deviates slightly from the published algorithm. For more details, see the section "Points of difference from the published algorithm" described [here](#).

Porter is approximately twice as fast as using [SnowballPorterFilterFactory](#).

Notes about solr.KStemFilterFactory

⚠ Solr3.3 KStem is an English language stemmer which is similar to Porter but less aggressive, and thus often preferred.

KStem is considerably faster than [SnowballPorterFilterFactory](#).

Notes about solr.SnowballPorterFilterFactory

Creates `org.apache.lucene.analysis.SnowballPorterFilter`.

Creates an [Snowball stemmer](#) from the Java classes generated from a [Snowball](#) specification. The language attribute is used to specify the language of the stemmer.

```
<fieldtype name="myfieldtype" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.WhitespaceTokenizerFactory" />
    <filter class="solr.SnowballPorterFilterFactory" language="German" />
  </analyzer>
</fieldtype>
```

Valid values for the language attribute (creates the snowball stemmer class language + "Stemmer"):

- [Armenian](#) ⚠ Lucene3.1
- [Basque](#) ⚠ Lucene3.1
- [Catalan](#) ⚠ Lucene3.1
- [Danish](#)
- [Dutch](#)
- [Kp](#): The Kraaij-Pohlmann stemming algorithm for Dutch.
- [Porter](#): The original Porter stemming algorithm for English.
- [English](#): The Porter2 stemming algorithm for English.
- [Lovins](#): The early Lovins stemming algorithm for English.
- [Finnish](#)
- [French](#)
- [German](#)
- [German2](#): A variation of the German algorithm with handling to allow ä, ö and ü to be represented by ae, oe and ue
- [Hungarian](#)
- [Italian](#)
- [Norwegian](#)
- [Portuguese](#)
- [Romanian](#)
- [Russian](#)
- [Spanish](#)
- [Swedish](#)
- [Turkish](#)

⚠ Gotchas:

- Although the Lovins stemmer is described as faster than Porter/Porter2, practically it is much slower in Solr, as it is implemented using reflection.

- Neither the Lovins nor the Finnish stemmer produce correct output (as of Solr 1.4), due to a [known bug in Snowball](#)
- The Turkish stemmer requires special lowercasing. You should use [TurkishLowerCaseFilter](#) instead of [LowerCaseFilter](#) with this language. See [background information](#).
- The stemmers are sensitive to diacritics. Think carefully before removing these with something like [ASCIIFoldingFilterFactory](#) before stemming, as this could cause unwanted results. For example, *r  sum  * will not be stemmed by the Porter stemmer, but *resume* will be stemmed to *resum*, causing it to match with *resumed*, *resuming*, etc. The differences can be more profound for non-english stemmers.

Customizing Stemming

Sometimes a stemmer might not do what you want out-of-box. For example, you might be happy with the results on average, but have a few particular cases (such as Product Names or similar) where it annoys you or actually hurts your search results.

The components below allow you to fine-tune the stemming process by preventing words from being stemmed at all, or by overriding the stemming algorithm with custom mappings.

`solr.KeywordMarkerFilterFactory`

 [Solr3.1](#)

Protects words from being modified by stemmers.

A customized protected word list may be specified with the "protected" attribute in the schema. Any words in the protected word list will not be modified by any stemmer in Solr.

A [sample Solr protwords.txt with comments](#) can be found in the Source Repository.

```
<fieldtype name="myfieldtype" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt" />
    <filter class="solr.PorterStemFilterFactory" />
  </analyzer>
</fieldtype>
```

`solr.StemmerOverrideFilterFactory`

 [Solr3.1](#)

Overrides stemming algorithms, by applying a custom mapping, then protecting these terms from being modified by stemmers.

A customized mapping of words to stems, in a tab-separated file, can be specified to the "dictionary" attribute in the schema. Words in this mapping will be stemmed to the stems from the file, and will not be further changed by any stemmer.

A [sample stemdict.txt with comments](#) can be found in the Source Repository.

```
<fieldtype name="myfieldtype" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.StemmerOverrideFilterFactory" dictionary="stemdict.txt" />
    <filter class="solr.PorterStemFilterFactory" />
  </analyzer>
</fieldtype>
```

Decompounding

Decompounding can improve search results for some languages. At the same time, it can increase the time it takes to index and search, as well as increase the index size itself.

Solr provides dictionary-based decompounding support via `solr.DictionaryCompoundWordTokenFilterFactory`. This factory allows you to provide a dictionary, along with some settings (min/max subword size, etc), to break compound words into pieces.

 [Solr3.1](#)

Additionally, you can use `solr.HyphenationCompoundWordTokenFilterFactory`. This factory uses a hyphenation grammar in combination with an optional dictionary to break compound words into pieces. Hyphenation grammars for a few languages can be found at the [FOP XML Hyphenation Patterns](#) site.

One alternative is to use n-gram tokenization so that the search is less sensitive to compound words.