Sukijan asennus- ja käyttöohje

Sukija on Javalla kirjoitettu ohjelma suomenkielisten tekstien indeksointiin.

Sukija analysoi sanat morfologisesti, muuttaa sanat perusmuotoon (joka on sanakirjoissa) ja indeksoi perusmuodot, jotta sanan kaikki taivutusmuodot löytyvät vain perusmuotoa etsimällä.

Sukija tallentaa perusmuodot Solr:n tietokantaan, josta niitä voi etsiä Solr:n käyttöliittymän kautta.

Sukija osaa indeksoida kaikkia niitä tiedostomuotoja, joita Apache Tika (http://tika.apache.org/) osaa lukea.

Mitä tarvitaan ja mistä ne saa?

- Sukija: `https://github.com/ahomansikka/sukija`
  Koska luet tätä tekstiä, olet jo imuroinut tämän. (-:
- Suomi-Malaga Se on corevoikossa (`https://github.com/voikko/corevoikko`) hakemistossa suomimalaga.
- Apache Solr 4.5.1: `http://lucene.apache.org/solr/`
- Ubuntun paketit `libmalaga7` ja `maven`.

Lisäksi Sukija tarvitsee erinäisiä jar-tiedostoja, mutta Maven imuroi ne verkosta automaagisesti.

Sukijaa voi käyttää myös Voikon Java-version kanssa. Tällöin tarvitaan Ubuntun paketti `libvoikko1`.

Tämä asennusohje olettaaa, että corevoikko ja apache-solr ovat hakemistoissa `$HOME/Lataukset/corevoikko/` ja `$HOME/Lataukset/solr/solr-4.5.1`

Jos ne ovat jossain muualla, tiedoston `Makefile` alussa olevia muuttujia `SOLR_HOME` ja `JETTY_CONTEXTS_DIR` pitää muuttaa vastaavasti.

Ohjelman rakenne

Sukijassa on kolme osaa:

- sukija-core Java-luokkia, joita muut ohjelman osat tarvitsevat.
- sukija-malaga Solr:n liitännäinen, joka käyttää Suomi-Malagan Sukija-versiota muuttamaan sanat perusmuotoon.
- sukija-voikko Solr:n liitännäinen, joka käyttää Voikkoa muuttamaan sanat perusmuotoon.

Suomi-Malagan asentaminen

Suomi-Malagasta on kaksi versiota, Voikko-versio on tarkoitettu oikolukuun ja Sukija tiedostojen indeksointiin. Sukija-versio käännetään komennolla

```
cd $HOME/Lataukset/corevoikko/suomimalaga
make sukija
```

Tee alihakemisto `$HOME/.sukija` ja kopioi sinne tiedostot
`suomimalaga/sukija/{suomi.*_l,suomi.pro}`

Myös Voikko-versiota voi käyttää indeksointiin, kun sen kääntää ja asentaa komennoilla

```
cd $HOME/Lataukset/corevoikko/suomimalaga
make voikko-sukija
make voikko-install DESTDIR=~/.voikko
```

`DESTDIR` voi olla myös joitan muuta kuin `~/.voikko`.

Versioiden erot ovat siinä, että Sukija-versio tunnistaa myös vanhoja taivutusmuotoja ja sanoja sekä yleisiä kirjoitusvirheitä.

Sukijan kääntäminen ja asentaminen, Solr:n konfigurointi

Ensin käännetään Sukija komennolla

`mvn package`

Komento imuroi netistä tarvitsemansa Javan jar-paketit eli ensimmäinen kääntäminen saattaa kestää kauan. Erityisen kauan se kestää, jos et ole aiemmin käyttänyt mavenia.

Toisessa vaiheessa asetetaan Solr:n konfigurointitiedostoon `schema.xml` saneistajaluokka (ulkomaankielellä tokenizer), joka lukee sanat tiedostoista, ja morfologialuokka, joka muuttaa sanat perusmuotoon. Komennolla `make ____-schema` on viisi eri vaihtoehtoa:

Komento	Morfologialuokka
`make malaga-schema`	MalagaMorphologyFilterFactory
`make malaga-suggestion-schema`	MalagaMorphologySuggestionFilterFactory
`make voikko-schema`	VoikkoMorphologyFilterFactory
`make voikko-suggestion-schema`	VoikkoMorphologySuggestionFilterFactory
`make debug-schema`	

Komentoa `make debug-schema` käytetään vain Sukijan kehittämiseen.

Saneistajan oletusarvo on FinnishTokenizerFactory, joka tulee Sukijan mukana, mutta sen voi vaihtaa muuttujalla TOKENIZER_FACTORY esimerkiksi näin:

```
make voikko-schema TOKENIZER_FACTORY=solr.StandardTokenizerFactory
```

`____FilterFactory` ja `____SuggestionFilterFactory` eroavat toisistaan siten, että jos morfologialuokka ei tunnista sanaa, `Suggestion`-luokissa sanaan tehdään muutoksia (esimerkiksi muutetaan w v:ksi) ja tunnistusta yritetään uudestaan. Tämä ei ole sama asia kuin Voikon oikeinkirjoituksen korjausehdotukset!

`Suggestion`-luokat konfiguroidaan tiedostossa `suggestion.txt`. Katso sivu 5.

Indeksoitavat tiedostot asetetaan tiedostossa `data-config.xml`.

Katso
`http://wiki.apache.org/solr/DataImportHandler#FileListEntityProcessor`
ja `http://wiki.apache.org/solr/TikaEntityProcessor`.

Tärkeimmät konfiguroitavat parametrit ovat:

- baseDir Hakemisto, jossa ja jonka alihakemistoissa tiedostot ovat.
- fileName Säännöllinen lauseke, joka valitsee indeksoitavat tiedostot.
- excludes Säännöllinen lauseke, joka valitsee tiedostot, joita ei indeksoida.

Komento `make install` asettaa näiden oletusarvoiksi

baseDir `$HOME/Asiakirjat`

fileName `.*` eli kaikki tiedostot indeksoidaan.

excludes
`(?u)(?i).*[.](au|bmp|bz2|class|gif|gpg|gz|jar|jpg|jpeg|m|o|png|tif|tiff|wav|zip)$`

Näitä voidaan muuttaa parametreilla `BASE_DIR`, `FILE_NAME` ja `EXCLUDES`. Esimerkiksi

```
make install BASE_DIR=/usr/local/data
```

Säännöllisten lausekkeiden syntaksi on sama kuin Javan luokassa
`java.util.regex.Pattern`.

Indeksoitavien tiedostojen asettamisen lisäksi komento `make install` kopioi hakemistossa `conf` olevat Solr:n ja Sukijan tarvitsemat tiedostot oikeisiin paikkoihin.

Tiedostot `data-config.xml`, `suggestion.txt` ja `synonyms.txt` kopioidaan hakemistoon `$HOME/.sukija` ja tiedostot `schema.xml`, `solrconfig.xml`,

`sukija-context.xml`, `sukija.xsl` ja alihakemisto `velocity` niihin hakemistoihin, joista Solr löytää ne.

Solr:n käynnistäminen

```
cd $HOME/Lataukset/solr/solr-4.5.1/example
java -jar start.jar
```

Jos Solr valittaa jna:sta, käynnistyskomento on

```
java -Djna.nosys=true -jar start.jar
```

Solr:n käynnistymisen voi varmistaa selaimessa menemällä verkko-osoitteeseen

`http://localhost:8983/solr/admin/`

Solr:n loki

Solr:n lokitulostus (`http://wiki.apache.org/solr/SolrLogging`) konfiguroidaan tiedostossa `solr-4.5.1/example/resources/log4j.properties` Mahdollisimman suuren lokitulostuksen saa lisäämällä tämän tiedoston loppuun rivit

```
log4j.logger.peltomaa.sukija.finnish.HVTokenizer = ALL
log4j.logger.peltomaa.sukija.morphology.MorphologyFilter = ALL
log4j.logger.peltomaa.sukija.suggestion.Suggestion = ALL
log4j.logger.peltomaa.sukija.suggestion.SuggestionFilter = ALL
log4j.logger.peltomaa.sukija.malaga.MalagaMorphology = ALL
log4j.logger.peltomaa.sukija.voikko.VoikkoMorphology = ALL
```

Tällöin tulostus on paljon suurempi kuin indeksoitavat tiedostot (-:, mutta kaikkia ei tietenkään tarvitse lisätä.

Lokitulostus menee tiedostoon `solr-4.5.1/example/logs/solr.log`

Indeksointi

Tiedostot indeksoidaan menemällä osoitteeseen

`http://localhost:8983/solr/dataimport?command=full-import`

Enemmän tai vähemmän pitkän ajan kuluttua indeksoinnin lopputuloksen voi katsoa osoitteesta

`http://localhost:8983/solr/dataimport`

Tietojen etsiminen

Sanoja etsitään menemällä osoitteeseen `http://localhost:8983/solr/browse`

Etsittävien sanojen tulee olla perusmuodossa. Etsittäessä sanoja ei muuteta perusmuotoon siksi, että yhden sanan perusmuoto voi olla toisen sanan taivutusmuoto. Paras esimerkki tästä on ”alusta”, joka on sanojen ”alusta”, ”alustaa”, ”alku”, ”alunen” ja ”alus” taivutusmuoto. Tällöin herää kysymys, mitä sanaa pitää etsiä, vai etsitäänkö kaikkia?

Tiedoston suggestion.txt konfigurointi

Tiedosto `suggestion.txt` pitää konfiguroida erikseen Sukijalle ja Voikolle. Nykyinen konfiguraatio on tehty Sukijalle.

Tässä vaiheessa kaikki indeksoitavista tiedostoista luetut sanat on muutettu pieniksi kirjaimiksi eli tiedostossa `suggestion.txt` olevat erisnimet pitää kirjoittaa pienellä alkukirjaimella.

Konfiguraatiossa on neljä komentoa. Komentojen nimet tulevat siitä, että ne on toteutettu samannimisinä Java-luokkina tai ainakin melkein: `Apostrophe` on toteutettu luokassa `ApostropheSuggestion` ja niin edelleen.

`Apostrophe` Poistaa sanasta heittomerkin ja yrittää tunnistaa sanan sen jälkeen. Jos tunnistaminen ei onnistu, poistaa sanasta heittomerkin ja kaikki sen jälkeiset merkit ja palauttaa jäljelle jääneet merkit sanan perusmuotona. Esimerkiksi yrittää tunnistaa merkkijonon `centime'in` muodossa `centimein`. Jos tunnistaminen ei onnistu, palauttaa merkkijonon `centime`.

`Char` Muuttaa sanassa olevan yhden merkin toiseksi. Esimerkiksi

```
Char "w" "v"
```

muuttaa w:t v:iksi (”wanha” => ”vanha”).

`CharCombination` Muuttaa yhden tai usemman merkin kaikki kombinaatiot. Esimerkiksi

```
CharCombination "pk" "bg"
```

(1) muuttaa p:t b:iksi, jättää k:t ennalleen, (2) muuttaa k:t g:iksi, jättää p:t ennalleen, sekä (3) muuttaa p:t b:iksi ja k:t g:iksi.

Siis jos tiedostosta luettu sana on ”piolokia”, komento yrittää tunnistaa sanat ”biolokia”, ”piologia” ja ”biologia”.

`Length3` poistaa kolmesta peräkkäisestä samasta kirjaimesta yhden.
Esimerkiksi ”kauttta” => ”kautta”.

`Regex` muuttaa säännöllisen lausekkeen. Esimerkiksi

```
Regex "ai(j)[eou]" ""
```

poistaa j-kirjaimen muun muassa sanoista ”aijemmin”, ”aijomme” ja ”kaijutin”.

Säännöllisessä lausekkeessa voi käyttää kirjainta `C` tarkoittamaan konsonantteja ja kirjainta `V` tarkoittamaan vokaaleja. Esimerkiksi

```
Regex  "C[ae](hi)C"  "i"
```

poistaa h-kirjaimet muun muassa sanoista ”ainahinen” ja ”etehinen”.

Kirjaimia `C` ja `V` lukuun ottamatta säännöllisten lausekkeiden syntaksi on sama kuin Javan luokassa `java.util.regex.Pattern`. Muita isoja kirjaimia ei tule käyttää säännöllisissä lausekkeissa, koska tässä vaiheessa kaikki tiedostoista luetut sanat on muutettu pieniksi kirjaimiksi, siis myös isokirjaimiset lyhenteet ja erisnimien alkukirjaimet.

Näitä komentoja voi antaa mielivaltaisen paljon missä tahansa järjestyksessä, ja `____SuggestionFilterFactory` palauttaa perusmuotona ensimmäisen tunnistamansa muodon. Jos mitään ehdotusta ei tunnisteta, `____SuggestionFilterFactory` palauttaa alkuperäisen merkkijonon.

Tiedostossa `suggestion.txt` voi olla tyhjiä rivejä. Kommentti alkaa merkillä `#` ja jatkuu rivin loppuun.