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 4.10.3: `http://lucene.apache.org/solr/`
  Tässä dokumentissa Solr:sta käytetään nimeä solr-x.y.z, missä x.y.z on version numero, esimerkiksi 4.10.3.
- 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-x.y.z`

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

Ohjelman rakenne

Sukijassa on neljä 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 (Malaga- tai Vfst-morfologiaa) muuttamaan sanat perusmuotoon.
- sukija-ui Javalla tehty käyttöliittymä (keskeneräinen).

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.

Testien aikana tulee virheilmoitus

SLF4J: Class path contains multiple SLF4J bindings.

Siitä ei tarvitse välittää.

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 vfst-schema`	VoikkoMorphologyFilterFactory
`make vfst-suggestion-schema`	VoikkoMorphologySuggestionFilterFactory
`make debug-schema`	
`make debug-vfst-schema`	

Vfst on Voikko, joka käyttää uutta vfst-morfologiaa.

Komentoja `make debug-schema` ja `make debug-vfst-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 `finnish-suggestion.xml`. Katso sivu 6.

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-x.y.z/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-x.y.z/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.SuccessFilter = ALL
log4j.logger.peltomaa.sukija.suggestion.SuggestionFilter = ALL
log4j.logger.peltomaa.sukija.malaga.MalagaMorphology = ALL
log4j.logger.peltomaa.sukija.voikko.VoikkoMorphology = ALL
log4j.logger.peltomaa.sukija.voikko.VoikkoMorphologySuggestionFilterFactory = ALL
log4j.logger.peltomaa.sukija.hyphen.HyphenFilter = ALL
```

Tuossa ovat kaikki Sukijan luokat, joissa on lokitulostus.

Tällöin tulostus on paljon suurempi kuin indeksoitavat tiedostot (-:, mutta kaikkia ei tietenkään tarvitse lisätä, riittää kun laittaa luokat `MorphologyFilter` sekä `SuggestionFilter` tai `SuccessFilter`.

Luokkia `SuggestionFilter` ja `SuccessFilter` ei pidä käyttää yhtaikaa.

Luokkaa `HVTokenizer` ei tarvitse laittaa, jos ei käytä tätä saneistajaa, ja luokkia `MalagaMorphology` ja `VoikkoMorphology` Sukija ei käytä yhtaikaa. `SuggestionFilter` tuottaa lokitulostuksen kaikista yrityksistä muuttaa sana perusmuotoon, `Suggestion` kertoo erikseen, mikä muunnos sai aikaan sanan tunnistamisen (katso sivu 6).

Lokitulostuksen eri tasot (`ALL`, jne) voi katsua luokan `org.apache.log4j.Level` dokumentoinnista.

Lokitulostus menee tiedostoon `solr-x.y.z/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?

Eri tavalla muotoillun tulostuksen saa osoitteesta
`http://localhost:8983/solr/select`

Esimerkiksi sanaa `sana` etsitään näin:

`http://localhost:8983/solr/select?q=sana`

Tämän tulostuksen ulkonäköä voi muuttaa muuttamalla Sukijan mukana tulevaa tiedostoa `conf/sukija.xsl`.

Tiedoston finnish-suggestion.xml konfigurointi

Tiedosto `finnish-suggestion.xml` pitää konfiguroida erikseen Sukijalle ja Voikolle. Nykyinen konfiguraatio on tehty Voikolle ja sen vfst-morfologialle.

Tässä vaiheessa kaikki indeksoitavista tiedostoista luetut sanat on muutettu pieniksi kirjaimiksi eli tiedostossa `finnish-suggestion.xml` olevat erisnimetkin pitää kirjoittaa pienellä alkukirjaimella.

Konfiguroititiedoston formaatti on määritelty tiedostossa `sukija-core/src/main/xsd/SuggestionInp`

Konfiguroinnissa on komennon nimi, jota voi seurata argumentteja, esimerkiksi

```
<suggestion name = "CompoundWordEnd">
  <argument>aukio aukio</argument>
</suggestion>
```

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

```
<suggestion name = "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 olevat merkit toiseksi. Tämä vastaa Unixin komentoa `tr`.

Esimerkiksi

```
<suggestion name = "Char">
  <argument>pk</argument>
  <argument>bg</argument>
</suggestion>
```

(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 ”piolokia”, ”biolokia”, ”piologia” ja ”biologia”.

```
<suggestion name = "Length3"/>
```

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

`CompoundWordEnd` tunnistaa yhdyssanan, jos se loppuu tiettyyn sanaan. Esimeriksi

```
<suggestion name = "CompoundWordEnd">
 <argument>aukio aukio</argument>
 <argument>jo(k[ie]|e) joki</argument>
</suggestion>
```

tunnistaa merkkijonot ”arkadianaukion” ja ”aatsajoelle”. Tällä tavalla voidaan tunnistaa paikannimiä, jotka eivät ole sanastossa.

Jokaisessa argumentissa on kaksi osaa. Ensimmäinen on säännöllinen lauseke ja toinen jonkin sanan perusmuoto. Tunnistettaessa merkkijono katkaistaan siitä kohdasta, josta säännöllinen lauseke alkaa, ja jos merkkijonon loppuosan perusmuoto on argumentin toinen osa, perusmuotona palauteaan merkkijonon alkuosa + argumentin toinen osa.

Esimeriksi ”aatsajoelle” jaetaan kahtia osiin ”aatsa” ja ”joelle”, ja koska merkkijonon ”joelle” perusmuoto on ”joki”, merkkijonon ”aatsajoelle” perusmuodoksi tulee ”aatsajoki”.

`Regex` muuttaa säännöllisen lausekkeen. Esimerkiksi

```
<suggestion name = "Regex">
  <argument>ai(j)[eou]  </argument>
  <argument>C[ae](hi)C  i</argument>
  <argument>true</argument>
</suggestion>
```

Ensimmäinen argumentti `Regex "ai(j)[eou]" ""` poistaa j-kirjaimen muun muassa sanoista ”aijemmin”, ”aijomme” ja ”kaijutin”.

Toinen argumentti `C[ae](hi)C i` poistaa h-kirjaimet muun muassa sanoista ”ainahinen” ja ”etehinen”

Viimeinen argumentti on joko `true`, jolloin kokeillaan kaikkia argumentteina olevia säännöllisiä lausekkeita ja palautetaan kaikki tunnistetut sanat, tai `false`, jolloin lopetetaan, kun ensimmäinen säännöllinen lauseke on tunnistettu.

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

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.

`String` muuttaa merkkijonon toiseksi. Esimerkiksi

```
<suggestion name = "String">
  <argument>tsh ts</argument>
  <argument>sydämm sydäm</argument>
</suggestion>
```

muuttaa merkkijonon ”mantshuria” merkkijonoksi ”mantsuria” ja merkkijonon ”sydämmellisesti” merkkijonoksi ”sydämellisesti”.

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.

Javalla kirjoitettu käyttöliittymä

Kun tiedostot on indeksoitu, niitä voidaan tutkia myös komennolla

```
java -jar sukija-ui/target/sukija-ui-1.0.jar
```