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 5.1.0: `http://lucene.apache.org/solr/` Tässä dokumentissa Solr:sta käytetään nimeä solr-x.y.z, missä x.y.z on version numero, esimerkiksi 5.1.0.
- 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 olevaa muuttujaa `SOLR` ja tiedoston `asenna.sh` alussa olevia muuttujia 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

Ensin käännetään ja asennetaan Sukija komennolla

`mvn install`

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

Sukijan jar-tiedostot asennetaan maven-hakemistoon `${HOME}/.m2`

Testien aikana tulee virheilmoitus

SLF4J: Class path contains multiple SLF4J bindings.

Siitä ei tarvitse välittää.

Solr:n konfigurointi (1)

Solr:n parametrit asetetaan tiedostossa `sukija.properties`, jossa on myös tarvittavat ohjeet.

Sen jälkeen kofiguroidaan Solr komennolla `make asenna`

Jos asentamisen jäkeen muuttaa tiedostoa `sukija.properties` tai `conf2/sukija-context.xml` muutokset voi viedä Solr:iin komennolla `make päivitä`

`make asenna` tekee saman kuin Solr:n komennot `bin/solr start; bin/solr create -c sukija -d conf`, mutta lisäksi se kopioi Solr:n tarvitsemat jar-tiedostot hakemistoon `$HOME/Lataukset/solr/solr-x.y.z/server/solr/sukija/lib`

Solr:n käynnistäminen

`make asenna` käynnistää Solr:n.

Myöhemmin sen voi käynnistää komennolla `~/Lataukset/solr/solr-x.y.z/bin/solr start`

Solr:n käynnistymisen voi varmistaa selaimessa menemällä verkko-osoitteeseen `http://localhost:8983/solr/`

Solr:n konfigurointi (2)

Tämä voidaan tehdä vasta sen jälkeen kun on tehty Solr:n konfigurointi (1).

Jos olet asentanut Solr:n skriptillä `install_solr_service.sh` (katso Apache Solr Reference Guide kohta Service Installation Script), Solr konfiguroidaan Sukijaa varten näin.

Skripti asentaa Solr:n hakemistoihin `/opt/solr-x.y.z` ja `/var/solr`. Niiden omistajaksi tehdään käyttäjä `solr`, mutta jos käytät lippua `-u`, voi asettaa hakemistojen omistajaksi itsesi:

```
sudo bash ./install_solr_service.sh solr-x.y.z.tgz -u username
```

missä username on oma käyttäjätunnuksesi. Nämä asennusohjeet olettavat, että olet tehnyt näin, että olet asentanut Solr:n oletushakemistoihin ja että olet Sukijan päähakemistossa eli hakemistossa, jossa on tiedosto ohje.tex.

Asennus konfiguroidaan komennolla `make service` joka yksinkertaisesti kopioi hakemiston `${HOME}/Lataukset/solr/solr-x.y.z/server/solr/sukija` hakemistoon `/var/solr/data` ja tiedoston `conf2/sukija-context.xml` hakemistoon `/opt/solr/server/contexts`.

Jos asentamisen jäkeen muuttaa tiedostoa `sukija.properties` tai `conf2/sukija-context.xml` muutokset viedään Solr:iin komennolla `make service-update`

Seuraavaksi mennään osoitteeseen `http://localhost:8983/solr/` Vasemmalla paneelin alalaidassa lukee

```
No cores available
Go and create one
```

Mennään sinne ja kirjotetaan kohtaan `name` ja `instanceDir` ”sukija” (mutta ilman lainausmerkkejä) ja napsautetaan kohtaa `Add Core`.

Nyt vasempaan paneeliin alalaitaan pitäisi tulla mahdollisuus valita indeksi (Solr käyttää siitä nimeä `core`) sukija. Sen jälkeen voimme ruveta indeksoimaan. Katso sivu 6.

Solr:n loki

Solr:n lokitulostus (`http://wiki.apache.org/solr/SolrLogging`) konfiguroidaan tiedostossa
`solr-x.y.z/server/resources/log4j.properties` tai
`/var/solr/log4j.properties`

Mahdollisimman suuren lokitulostuksen saa lisäämällä tämän tiedoston loppuun rivit

```
log4j.logger.peltomaa.sukija.finnish.HVTokenizer = ALL
log4j.logger.peltomaa.sukija.hyphen.HyphenFilter = ALL
log4j.logger.peltomaa.sukija.malaga.MalagaMorphology = ALL
log4j.logger.peltomaa.sukija.voikko.MalagaMorphologyFilterFactory = 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.suggestion.SuggestionParser = ALL
log4j.logger.peltomaa.sukija.voikko.VoikkoMorphology = ALL
log4j.logger.peltomaa.sukija.voikko.VoikkoMorphologyFilterFactory = 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 luokan `MorphologyFilter` tai `SuggestionFilter`.

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
`$HOME/Lataukset/solr/solr-x.y.z/server/logs/solr.log` tai
`/var/solr/logs/solr.log`

Indeksointi

Tiedostot indeksoidaan menemällä osoitteeseen
`http://localhost:8983/solr/sukija/dataimport?command=full-import`

Enemmän tai vähemmän pitkän ajan kuluttua indeksoinnin lopputuloksen voi katsoa osoitteesta `http://localhost:8983/solr/sukija/dataimport`

Tietojen etsiminen

Sanoja etsitään menemällä osoitteeseen
`http://localhost:8983/solr/sukija/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/sukija/select`

Esimerkiksi sanaa `sana` etsitään näin:
`http://localhost:8983/solr/sukija/select?q=sana`

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

Tiedoston suggestions.xml konfigurointi

Tiedosto `suggestions.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 `suggestions.xml` olevat erisnimetkin pitää kirjoittaa pienellä alkukirjaimella.

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

Konfigurointitiedostossa olevien säännöllisten lausekkeiden syntaksi on sama kuin Javan luokassa `java.util.regex.Pattern`

Konfiguroinnille ei ole käyttöliittymää, koska siinä voi käyttää mitä tahansa XML-editoria.

`compoundWordEnd` tunnistaa yhdyssanan, jos se loppuu tiettyyn sanaan. Esimerkiksi

```
<compoundWordEnd>
  <input>jo(k[ie]|e) joki</input>
</compoundWordEnd>
```

tunnistaa esimerkiksi merkkijonon ”aatsajoelle”. Tällä tavalla voidaan tunnistaa paikannimiä, jotka eivät ole sanastossa.

Jokaisessa input-lauseessa 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”.

Input-lauseita voi olla mielivaltainen määrä.

`prefix` tunnistaa etuliitteettömän sanan (”etuliite” voi olla mikä tahansa merkkijono). Esimerkiksi

```
<prefix>
  <prefix>abcdefg</prefix>
  <savePrefix>true</savePrefix>
  <saveWord>true</saveWord>
</prefix>
```

poistaa sanan alusta merkkijohon ”abcdefg” ja yrittää tunnistaa jäljelle jääneen sanan (”abcdefgsuomalaiselle” => ”suomalaiselle”) ja tallentaa sen perusmuodon (”suomalainen”). Jos `savePrefix` on true, tallentaa myös etuliitteen (”abcdefg”) ja jos `saveWord` on true, tallentaa myös koko sanan perusmuodon (”’abcdefgsuomalainen”).

`char` muuttaa sanassa olevat merkit toiseksi. Tämä vastaa Unixin komentoa `tr`. Esimerkiksi

```
<char>
  <from>gbdkptvw</from>
    <to>kptgbdwv</to>
</char>
```

muuttaa g:n k:ksi, b:n p:ksi jne. Ohjelma testaa muutettavien kirjainten kaikki mahdolliset kombinaatiot. Esimerkiksi jos tiedostosta luettu sana on ”piolokia”, komento yrittää tunnistaa sanat ”piolokia”, ”biolokia”, ”piologia” ja ”biologia”.

`regex` muuttaa säännöllisen lausekkeen. Esimerkiksi

```
<regex>
 <input>(ai)(j)([eou])   $1$3</input>
 <input>^([0-9]+)</input>
 <tryAll>true</tryAll>
</regex>
```

Ensimmäinen input-lause poistaa j-kirjaimen muun muassa sanoista ”aijemmin”, ”aijomme” ja ”kaijutin”, ja toinen poistaa numerot sanan alusta.

Säännöllisessä lausekkeessa voi käyttää seuraavia lyhenteitä:

```
%A  [aä]
%C  [bcdfghjklmnpqrsštvwxzž]
%O  [oö]"
%U  [uy]"
%V  [aeiouyäö]
%%  %
```

Esimerkiksi `<input>(%V)(h)(%V)  $1hd$3</input>` muuttaa h:n hd:ksi esimerkiksi sanassa ”puhistus”.

Jos `tryAll` on true, ohjelma kokeilee kaikkia säännöllisiä lausekkeita, jos se on false, ohjelma lopettaa ensimmäisen tunnistetun sanan jälkeen.

Input-lauseen ensimmäinen osa on säännöllinen lauseke ja toinen merkkijono, miksi se muutetaan. Sen syntaksi on sama kuin Javan luokassa `java.util.regex.Matcher`.

Jos toista osaa ei ole, säännöllisen lausekkeen tunnistama merkkijono poistetaan. Input-lauseita voi olla mielivaltainen määrä.

`start` käy läpi kaikki sanan alut pisimmästä (maxLength) alkaen lyhimpään (minLength) asti ja lopettaa, kun löytyy ensimmäinen tunnistettu sana

```
<start>
  <minLength>4</minLength>
  <maxLength>10</maxLength>
  <baseFormOnly>true</baseFormOnly>
</start>
```

Jos `baseFormOnly` on true palautetaan sana vain, jos se on perusmuodossa (esim ”autowerwwwww” palauttaa ”auto”), muuten palautetaan tunnistettu sana muutettuna perusmuotoon (”kuudenwwwww” palauttaa ”kuusi”, mutta jos `baseFormOnly` on false, ei palauteta mitään).

`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`.

```
<apostrophe/>
```

Tällä komennolla ei ole parametreja.

Javalla kirjoitettu käyttöliittymä

Kun tiedostot on indeksoitu, niitä voidaan tutkia myös komennolla

```
java -jar $HOME/.m2/repository/peltomaa/sukija/sukija-ui/1.1/sukija-ui-1.1.jar
```