Sukijan asennus- ja käyttöohje

Sukija on Javalla kirjoitettu Solr:n liitännäinen 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 ja tekstitiedostojen merkistökoodauksia, 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. (-:
- Corevoikko: `https://github.com/voikko/corevoikko` Uusin versio. C-koodin pitää olla käännetty niin, että sanojen perusmuodot tulevat mukaan.
- Solr 9.9.0: `http://lucene.apache.org/solr/` Tässä dokumentissa Solr:sta käytetään nimeä solr-x.y.z, missä x.y.z tarkoittaa version numeroa.
- Javan versio 17 tai uudempi. Sukija ei ehkä käänny Javan aiemmilla versiolla.
- Apache mavenin versio 3.8.4 tai uudempi. Aiemmat mavenin versiot eivät tue Java 17:ää.
- Apache Lucene ja Apace Tika. Niiden pitää olla samoja versioita, joita solr-x.y.z käyttää. Tarkista asia tiedostossa pom.xml.

Lisäksi Sukija tarvitsee erinäisiä jar-tiedostoja, mutta Maven imuroi kaikki Sukijan tarvitsemat jar'it automaagisesti.

Tämä asennusohje olettaa, että libvoikko on asennettu hakemistoon `/usr/local/lib` ja että Solr on hakemistossa `$HOME/Lataukset/solr/solr-x.y.z`

Jos ne ovat jossain muualla, tiedostossa `sukija.properties` olevia Libvoikon tietoja pitää muuttaa vastaavasti.

Solr:n asentaminen

Solr:ia ei tarvitse asentaa, vaan sitä voi käyttää suoraan hakemistosta `$HOME/Lataukset/solr/solr-x.y.z`.

Tuotantokäyttöä varten Solr on parasta asentaa palveluna (service). Tämä ohje olettaa, että Solr on asennettu palveluna.

Katso

```
https://solr.apache.org/guide/solr/latest/
Deployment Guide -> Installation & Deployment
 -> Taking Solr to Production
```

Jos kone on kiinni verkossa, kannattaa katsoa myös ohjeet turvallisuudesta

```
https://solr.apache.org/guide/solr/latest/
Deployment Guide -> Securing Solr
```

Mitä turvallisuuteen tulee, tiedostossa `/etc/default/solr.in.sh` oleva rivi

```
#SOLR_SECURITY_MANAGER_ENABLED=true
```

pitää muuttaa muotoon

```
SOLR_SECURITY_MANAGER_ENABLED=false
```

tai muuten Solr ei osaa ladata Voikon binaaritiedostoa libvoikko.so.

Älkää säikähtäkö: Java 17:ssä SecurityManager on merkitty vanhentuneeksi (ulkomaankielellä ”deprecated”), ja se poistetaan jossain Javan tulevassa versiossa.

Mielestäni latauksen pitäisi toimia, jos Solr:n tiedostoon `server/etc/security.policy` lisää rivin

```
permission java.lang.RuntimePermission "loadLibrary.jnidispatch";
```

mutta se ei toimi. Jos saatte homman toimimaan muuttamatta tiedostoa `/etc/default/solr.in.sh` kertokaa minulle.

Lisäksi samaan tiedostoon pitää lisätä rivi

```
SOLR_CONFIG_LIB_ENABLED=true
```

tai muuten tiedostossa `solrconfig.xml` olevat `<lib ... />` -komennot eivät enää toimi.

Sukijan kääntäminen ja asentaminen

Sukija käännetään ja asennetaan 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-tiedosto asennetaan maven-hakemistoon `${HOME}/.m2`

Sukijan konfigurointi

Sukijan parametrit asetetaan tiedostossa `sukija.properties`, jossa on myös tarvittavat ohjeet.

Sen jälkeen komennolla

```
make tiedostot
```

tehdään Solr:n asetustiedostot `conf/indexer-config.xml` (katso sivu 6) ja `conf/schema.xml`. Esimerkit näistä tiedostoista saa aikaan yllä mainitulla komennolla. (-:

Seuraavaksi pitää tehdä Solr:iin Sukijan tarvitsemat hakemistot ja kopioida niihin Solr:n tarvitsemat tiedostot. Sitä varten on kaksi skriptiä

```
src/main/resources/install.sh
src/main/resources/install-sudo.sh
```

Jälkimmäistä käytetään, jos Solr on asennettu tunnuksella `solr`.

Jos asentamisen jälkeen konfiguroi Sukijaa, muutokset voi viedä Solr:iin jommallakummalla komennoista

```
cp -r conf/* /var/solr/data/sukija/conf/
sudo -u solr cp -r conf/* /var/solr/data/sukija/conf/
```

Sukijan jar-paketin voi päivittää näin

```
cp $HOME/.m2/repository/peltomaa/sukija/sukija/3.0.0/sukija-
3.0.0.jar /var/solr/data/sukija/lib/
```

Kaikki pitää laittaa yhdelle riville ja mahdollisesti lisätä `sudo` komennon alkuun.

Päivityksen jälkeen Solr pitää käynnistää uudellen. Linuxissa se käy näin

```
sudo service solr restart
```

Jos Solr:ia käyttää selaimen kautta, sekin on syytä käynnistää uudelleen.

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

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

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

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

Sukijan loki

Sukijan lokitulostus konfiguroidaan Solr:n lokitulostuksen kautta tiedostossa `/var/solr/log4j2.xml`. Sukijan käyttämät lisäykset ovat on hakemistossa `src/main/resources/log4j2.xml`. Ne on kommentoitu pois, mutta niitä voi siirtää halutun määrän kommenttien ulkopuolelle. Jotta Solr näkisi muutokset, ne pitää laittaa tiedostoon `/var/solr/log4j2.xml`.

Lokitulostuksen eri tasot (`all`, jne.) voi katsoa Solr:n dokumenteista.

Lokitulostus menee kansioon `/var/solr/logs/`

Indeksointi

Versiosta 9.0.0 alkaen Solr:sta on poistettu `DataImportHandler`, jota käytin kovalevyni indeksointiin, ja sen korvaajaksi tein ohjelman `src/main/resources/indexer.sh`

Se toimii minulla, mutta en väitä, että se toimii sinulla.

`src/main/resources/indexer.sh` konfiguroidaan tiedostossa `sukija.properties`. Katso sivu 4. Tiedoston `conf/indexer-config.xml` kentät ovat (nimet on kopioitu `DataImportHandler`ista):

core	Sukijan indeksin URL-osoite.
file	Säännöllinen lauseke, joka kuvaa indeksoitavien tiedostojen nimet.
excludes	Säännöllinen lauseke, joka kuvaa niitten tiedostojen nimet, joita ei indeksoida.
tika	Apache Tikan asetustiedosto, jota Sukija käyttää.
writeLimit	Indeksoitavan tiedoston maksimikoko. -1 tarkoittaa, että ylärajaa ei ole. Jos writeLimit > 0 indeksoidaan tiedoston alusta enintään niin monta tavua ja lopetetaan.
commitWithinMs	Enimmäisaika millisekunteina ennen kuin indeksoitavat tiedostot tallenetaan Solr:ään.
onError	Mitä tehdään virheilmoituksen jälkeen? `abort` lopettaan. `skip` siirrytän indeksoimaan seuraavaa tiedostoa.
recursive	Indeksoidaanko baseDir-hakemistojen alihakemistot (`true`, `false`)?
baseDir	Indeksoitavat hakemistot, erottimena `${path.separator}`, esimerkiksi `${user.home}/Asiakirjat:/usr/local/data` Linuxissa erottimena on kaksoispiste : Windowsissa erottimena taitaa olla puolipiste ;

Kenttien oletusarvot ovat tiedostossa `sukija.properties`

Myös `Apache Nutch`in voi konfiguroida indeksoimaan kovalevyn eli paikallisen tiedostojärjestelmän.

`src/main/resources/indexer.sh` ei toimi (ainakaan minulla) Solr:n versiossa 9.9.0, vaan se yrittää joskus indeksoida tekstitiedostoja, joitten

MIME-tyyppi on `text/plain` tai `text/x-tex` ikään kuin ne olisivat jotain Microsoftin formaattia.

Mutta ei hätää, olen tehnyt ohjelman

`src/main/resources/indexer2.sh`

joka käyttää samaa asetustiedostoa kuin `indexer.sh` ja toimii ainakin minulla. En väitä, että se toimii sinulla.

Tietojen etsiminen

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

Etsittävien sanojen tulee olla perusmuodossa (yksikön nominatiivissa) ja pienellä kirjoitettuna (myös erisnimet). 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?

Myös monikolliset sanat on tallennettu yksikössä. Esimerkiksi Yhdysvaltojen perusmuoto on Yhdysvalta.

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`.

Tietoja voi etsiä myös Solr:n käyttöliittymällä kohdassa `Query`.

Koska sanat on tallennettu perusmuodossa, fraasihaussakin sanojen täytyy olla perusmuodossa. Esimerkiksi Helsingin yliopistoa pitää etsiä fraasihaulla

`"helsinki yliopisto"`

Indeksoinnissa w:t muutetaan v:ksi eli esimerkiksi Waltaria on etsittävä muodossa

`valtari`

Tiedoston suggestions.xml konfigurointi

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

Konfigurointitiedoston formaatti on määritelty tiedostossa `/src/main/xsd/SuggestionInput.xsd`.

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

Konfiguroinnille ei ole käyttöliittymää, koska tiedoston `suggestions.xml` muokkaamiseen voi käyttää mitä tahansa XML-editoria.

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

```
<compoundWordEnd>
  <input>joki joki</input>
  <input>joke joki</input>
  <input>joe joki</input>
</compoundWordEnd>
```

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

Jokaisessa input-lauseessa on kaksi osaa. Ensimmäinen on jokin merkkijono ja toinen jonkin sanan perusmuoto. Tunnistettaessa merkkijono katkaistaan siitä kohdasta, josta ensimmäinen merkkijono alkaa, ja jos merkkijonon loppuosan perusmuoto on argumentin toinen osa, perusmuotona palautetaan 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.

Ensimmäinen merkkijono ei ole säännöllinen lauseke, koska ne ovat hitaampia kuin merkkijonot varsinkin, kun merkkijonojen tunnistuksessa

käytetään Aho-Corasick -algoritmia.[1] Algoritmissa sanan (ensimmäisen merkkijonon) etsimiseen käytetty aika ei riipu etsittävien sanojen määrästä, sillä se etsii kaikkia sanoja samanaikaisesti.

`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 merkkijonon ”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”).

Prefix-lauseita voi olla mielivaltainen määrä.

`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” (mutta ei välttämättä tässä järjestyksessä :-).

`regex` muuttaa säännöllisen lausekkeen. Esimerkiksi

[1] Katso Alfred V. Aho and Margaret J. Corasick: *Efficient String Matching: An Aid to Bibliographic Search.* Communications of the ACM. June 1975 Volume 18 Number 6. Löytyy netistä googlaamalla.

```
<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.