Using SALAMA (Swahili Language Manager) in corpus work

Arvi.Hurskainen@helsinki.fi

0. Introduction

These instructions are for those who have permission to use SALAMA in working with Swahili corpus.

If you are not acquainted with the unix/linux environment, you should first learn at least the basics of how to work in this environment. Such info can be found in the following addresses:

http://www.routledge.com/linguistics/introduction.html
Read especially Chapter 5.

http://www.aakkl.helsinki.fi/cameel/corpus/rmecorp.htm

Instructions given in the latter address above contain basic information on unix/linux and how to use Swahili corpus with general tools.

SALAMA is a sophisticated approach to corpus work and it facilitates accurate and comprehensive management if information. The idea behind this approach is that we first carry out linguistic analysis of the text, and then we make use of the analysis result while working with corpus text. In some sense this approach is similar with working with a tagged corpus. There are significant differences, however. While in working with a tagged corpus we have access to a text that is already provided with morphological and perhaps also other kinds of tags. In the approach presented here, tagging is done in flight, so to speak, and the user does not necessarily even notice that he or she is in fact dealing with a tagged corpus.

The description of tags used in SALAMA is available in

Tagset of SWATWOL

The reasons for not providing the user with a readily tagged corpus are manifold:
(1) The tagged corpus takes space at least ten times more than the raw text. It is not always possible to maintain such huge text files in the corpus server.

(2) The user can do the tagging with the programs provided in SALAMA, if a tagged version of the corpus is needed.

(3) There are several levels and modes of tagging, depending on the need the corpus is used for. If we would provide a readily tagged corpus, we in fact would need to provide several versions of the tagged corpus, which would require still more space. The user can use the untagged corpus directly also for such tasks which require plain text instead of tagged text.

(4) The work with untagged corpus is vastly more flexible than the work with a corpus that already has been tagged. These points will be elaborated below.

1. The processing sequence

Texts in Swahili corpus are available in two formats. Some texts are just plain ASCII text without any kind of pre-processing. Most texts, however, are in the pre-processed format. When running the programs described below you should always check the format of the text, so that you know what version of the program you should use. The rule is:

If your source text is in pre-processed format, you should use a program with the \texttt{-snt} extension. The programs that expect raw text as input do not have such an extension in the name.

1.1. Programs that expect raw text as input

The basic processing sequence is described below. The name of the program is given and its function is described briefly.

1. \texttt{pre-process}
   It carries out a number of operations to the text, the purpose of which is to make the text optimal for linguistic analysis. The program does not tag the text in any way.

2. \texttt{analyze}
   It carries out the basic linguistic analysis of the text. There are several versions of the analyzer. The default is that the analyzer carries out full morphological analysis and gives also English glosses to the words. It also treats verbal extensions as separate words, so that each extended verb has the extended stem as base form, or lemma.

2.1. \texttt{analyze-simple-lemma}
   It carries out the morphological analysis as \texttt{analyze} does, but for extended verbs it gives the
non-extended base form as lemma.

2.2. analyze-no-glosses
It carries out the morphological analysis as analyze does, but it leaves the English glosses out.

2.3. analyze-no-glosses-simple-lemma
It functions as analyze, but leaves English glosses out and gives only the simple verb base as lemma for extended verbs.

2.4. analyze-only
It performs the morphological analysis only. The input text should be pre-processed. (It functions as analyze-snt, listed below, but for logical reasons it is also here, with a different name).

2.5. analyze-only-simple-lemma
It performs morphological analysis only but gives as output a simple verb base as lemma for extended verbs.

3. disambiguate
It resolves ambiguous analyses of words. It tries to leave only one analysis for each word. However, there are words, for which the disambiguator is not able to find solution (yet).

3.1. disambiguate-only
It resolves ambiguity in morphological analysis just as disambiguate does. The difference in the function is that disambiguate-only expects that the text has already been analyzed. You should not try to run this for raw text!

3.2. disambiguate-simple-lemma
It resolves ambiguous analyses of words. It functions as disambiguate, but gives as output a simple verb base as lemma for extended verbs.

4. one-line-format
It moves the word-form and its analysis to the same line. In this format it is easy to keep each word with its analysis together. Sorting and counting lines is possible without mixing words and their analysis.

4.1. one-line-format-simple-lemma
It functions as one-line-format but gives as output a simple verb base as lemma for extended verbs.

4.2. one-line-format-only
It puts the word and its analysis on the same line. It expects that the text has already been analyzed and disambiguated. Do not try to analyze raw text with this command!
5. list-count-lemmas
It gives only the frequencies of each lemma in text. Note that when using this option, the ambiguity of word-forms is not taken into account. A word with different interpretations is counted as a single word.

5.1. list-count-lemmas-analyze
It gives the frequencies of each lemma in text but also some morphological information, as well as English glosses. This format suits for dictionary compilation.

5.2. list-count-lemmas-simple
It gives only the frequencies of each lemma in text and gives a simple base form for extended verbs as lemma.

5.3. list-count-lemmas-analyze-simple
It gives only the frequencies of each lemma in text and gives a simple base form for extended verbs as lemma. It also gives some morphological information, as well as English glosses. This format suits for dictionary compilation.

Note that all the programs (1-6) expect raw text as input, of no other information is given. In other words, each program performs the whole sequence of analysis starting from pre-processing. You do not need to put different programs into the pipe yourself. For example, the program called disambiguate performs the phases (1 - 3), the program called list-count-lemmas performs the phases (1 - 5), and so on.

Note, however, that the programs with the sequence -only in name perform a restricted sequence of commands, normally only one operation. Use these programs with care, because, if wrongly used, you get error messages. They are meant for more experienced users who understand the whole process of analysis. By using these commands you may build your own sequences of commands, by putting them into a pipe. For example, the command:

venus$ cat test.txt | /corp/swa/bin/pre-process | /corp/swa/bin/
analyze-only-simple-lemma | /corp/swa/bin/disambiguate-only | /corp/
swa/bin/one-line-format-only > test.res

performs the same operation as:

venus$ cat test.txt | /corp/swa/bin/one-line-format-simple-lemma >
test.res

1.2. Filter programs
There are also some filters which remove such information that is not needed in a particular task.

- **prune-tags**: It removes such tags that are not needed in compiling a dictionary.
- **remove-num**: It removes numbers found in text.

These filter programs should be **run after the disambiguation of text** has been carried out. They should not be run earlier, because the disambiguation rules need the full text and full analysis to work satisfactorily. You may consider using these filter programs as post-processing, as you use such Linux filters as `sort`, `uniq`, `rev`, etc.

### 1.3. Programs that expect the source text to be in pre-processed format

Each of the programs (1 - 6) has a counterpart program, which expects that the source text is already pre-processed. This is to help the user, because part of the corpus texts is available only in pre-processed format, while other texts are in original format. You should first check which format the text has, before you try to run a program.

Such programs are:

- `analyze-snt`
- `analyze-simple-lemma-snt`
- `analyze-no-glosses-snt`
- `analyze-no-glosses-simple-lemma-snt`
- `disambiguate-snt`
- `disambiguate-simple-lemma-snt`
- `one-line-format-snt`
- `one-line-format-simple-lemma-snt`
- `list-count-lemmas-snt`
- `list-count-lemmas-analyze-snt`

Note that each program that expects **pre-processed text** as input has the extension `-snt`. 
You see above in (1.1. 1-6) what these programs do.

2. Hints on use

Hint 1:
The programs described above are not available to all users of the corpus server. Therefore, while using the programs, you should specify the full path of the programs. The programs described here are located in the directory /corp/swa/bin/.

venus@ cat /corp/swa/standard/articles/alasiri/ala-all | /corp/swa/bin/one-line-format > ala-all.r

This command analyzes the text, disambiguates it, puts the result into one line format, and finally directs the result into a file called ala-all.r.

Hint 2:
Be careful not to create too big files. If you need such files, you should quickly move them to your own computer. Recall that even a disambiguated file is ten times bigger than the original file.

There are methods to handle the size problem:

1. Zip the file by using

venus$ gzip file

The resulting condensed file will have the format file.gz.

You can unzip the file with the command

venus$ gunzip file.gz

The file gets its original form.

You can also use the zipped file without unzipping it. The file is opened in flight.

venus$ zcat file.gz | commands...

2. If you do not need the analyzed words in the order where they are in text, you can down-size the resulting file considerably by sorting and removing duplicate lines.
Hint 3:
It is wise to test the function of the program, or a sequence of programs, first. A handy method of testing is to to use unix commands head and tail for getting ten first or last lines of the file. When you are sure that the program does what you want, you can process larger amounts of text. For example, if you want to see whether the file books-all is pre-processed or not, you may invoke the command:

You can then make a test with a program. For example:

```
[1039] venus$ head /corp/swa/standard/books/books-all
<FAS> *makala za *semina ya *kimataifa ya *waandishi wa *kiswahili
*iii *fasihi *taasisi ya *uchunguzi wa *kiswahili *chuo *kikuu cha
*dar_es_*salaam , 1983
<FAS> *dibaji $
<FAS> *kitabu hiki , ambacho kinatolewa katika juzuu tatu , ni matokeo
ya semina mbili za kimataifa za waandishi wa *kiswahili .$
<FAS> *semina ya kwanza ilifanyika *dar_es_*salaam , *tanzania ,
kuanzia tarehe 16 / 9 / 78 hadi 7 / 10 / 78 .$
<FAS> *semina ya pili vilevile ilifanyika *dar_es_*salaam kuanzia
<FAS> *semina zote mbili zilisimamiwa na *taasisi ya *uchunguzi wa
*kiswahili , *chuo *kikuu cha *dar_es_*salaam , na kugharamiwa na
*shirika la *kimataifa la *maendeleo la *sweden ( s*i*d*a ) kupitia
*shirika la *umoja wa *mataifa la *elimu *sayansi na *utamaduni
( u*nesco )$
<FAS> *washiriki wa semina zote mbili walitoka katika nchi za *afrika
*mashariki na *kati zenye wazungumzaji wa *kiswahili , *nchi
zilizoalikwa ni *burundi , *kenya , *ethiopia , *malagasi , *malawi ,
*msumbiji , *ngazija , *rwanda , *somalia , *sudan , *tanzania ,
*uganda na *zaire .$
<FAS> *zote , isipokuwa *ethiopia , *malawi , *ngazija na *zaire ,
zililetwa wajumbe wao .$
<FAS> *kadhalika , semina zilihudhuriwa na watajamaji kutoka *sweden
( s*i*d*a ) na u*nesco .$
<FAS> *semina zote mbili zilikuwa na shabaha za jumla zifuatazo :$
```

It is a pre-processed text. Then you may make a test with a program.

```
[1040] venus$ head /corp/swa/standard/books/books-all | /corp/swa/bin/
one-line-format-snt
"<<FAS>>" &
Hint 4:
We will make another test, now with a newspaper file.

[1051] venus$ head /corp/swa/standard/articles/alasiri/ala-all
[ Alasiri 26th *january 1999 ] ' *bomu ' jengo la *p*p*f laitikisa
*dar lwanajeshi , polisi walizingira *jumanne , *januari 26,1999 *na
*agnether *kasenene , *jijini *hofu , kiwewe na mshikemshike leo
zilitawala katikati ya *jiji hasa maeneo ya mitaa ya *samora na ile ya
jirani kutokana na taarifa zilizotolewa polisi kwamba , kuna bomu
limetegwa kwenye jengo jipya la *p*p*f na lingelipuliwa saa nne
kamili .
*jengo hilo lenye ghorofa 14 linalomilikiwa na *mfuko wa *pensheni wa
*mashirika ya *umma , lijulikanalo kama *p*p*f *house , lipo kwenywe
makutano ya barabara za *samora na *morogoro , jirani kabisa na ofisi
za *tume ya *jiji .
*kamanda wa *polisi wa *mkoa wa *dar es *salaam , *bw .
*alfred *gewe amesema kuwa baada ya kupewa taarifa hizo kuwa kuna
bomu , alimwaga *askari *wataalam wa masuala hayo , *polisi wa kawaida
na *askari wa *jeshi la *wananchi *tanzania kwa ajili ya kupambana na
Using SALAMA (Swahili Language Manager) in corpus work

hali hiyo.
*askari hao wamesema kuwa walipofika kwenyeni tukio, waliwazuia wafanyakazi na wapangaji wote wa jengo hilo kuingia.
*wamesema ilibidi wapekue jengo zima chumba hadi chumba kukagua kama kuna bomu ili walitegue kama watalikutu.
*baadhi ya watu wamesema sakata hilo lilianza jana baada ya mfanyabiashara mmoja kutoa taarifa kituo cha *polisi cha *kati kuwa, ana wasiwasi kuna bomu limetegwa kwenyeni jengo hilo.
*alipohojiwa na gazeti hili, mfanyabiashara huyo mwenye ofisi ya uwakala wa *bima amekiri kutoa taarifa hizo polisi.
*mfanyabiashara huyo, ofisi ya kampuni yake iko ghorofa ya tano jina lake halisi ni *bw.
*sanjay *suchak.

It is also a pre-processed text.

The following sequence of commands puts the result into the word-per-line format, sorts it alphabetically, then deletes duplicate lines but retains count, and this result is then sorted according to number and arranged so that the biggest number is in the beginning of the file. The larger the source file is, the bigger the advantage of this kind of post-processing is.

```
[1048] venus$ cat /corp/swa/standard/articles/alasiri/ala-all | /corp/swa/bin/one-line-format | sort | uniq -c | sort -nr > ala-all.rsn
```

```
19554"<,>" COMMA &
8230 "<na>" "na" CC @CC ' and ' &
5901 "<ya>" "a" GEN-CON 9/10-SG &
3724 "<kwa>" "kwa" PREP ' at, to, for ' &
3128 "<wa>" "a" GEN-CON 1/2-SG &
3101 "<kuwa>" "kuwa" CONJ **CLB ' that ' &
2854 "<" DOUBLE-QUOTE &
2673 "$i" DEF-V:ni ' be ' " SG1-SP &
2495 "<katika>" "katika" PREP ' in, at ' &
2342 "<za>" "a" GEN-CON 9/10-PL &
2233 "<la>" "a" GEN-CON 5/6-SG &
2131 "<hiyo>" "hi-o" PRON DEM :hV ASS-OBJ 9/10-SG ' this ' &
2119 "<wa>" "a" GEN-CON 11-SG &
2070 "<ya>" "a" GEN-CON 5/6-PL &
2051 "<wa>" "a" GEN-CON 1/2-PL &
2016 "<cha>" "a" GEN-CON 7/8-SG &
1718 "<na>" "na" AG-PART ' by ' &
1478 "<huyo>" "hu-o" PRON DEM :hV ASS-OBJ 1/2-SG' this ' &
1453 "<na>" "na" PREP ' with ' &
1413 "<' DOUBLE-QUOTE &
1285 "<kama>" "kama" ADV AR ' like , such as ' &
```
Hint 5:

If we want to sort the result according to the lemma, we use the following sequence of commands. Here again, only a small extract is given.

```
venus$ cat /corp/swa/standard/articles/alasiri/ala-all | /corp/swa/bin/one-line-format | sort | uniq -c | sort +2 > ala-all.rsl
 1  "<waliandikisha>" "andikisha"  V 1/2-PL3-SP VFIN PAST  SV SVO  ' SVOO  ' write  ' CAUS &
 5  "<waliandikisha>" "andikisha"  V 1/2-PL3-SP VFIN PAST 1/2-PL-REL REFL-SG-OBJ SV SVO  ' SVOO  ' write  ' CAUS &
 1  "<waliandikisha>" "andikisha"  V 1/2-PL3-SP VFIN PAST 3/4-SG-REL REFL-SG-OBJ SV SVO  ' SVOO  ' write  ' CAUS &
 2  "<waliandikisha>" "andikisha"  V 1/2-PL3-SP VFIN PERF:me REFL-SG-OBJ SV SVO  ' SVOO  ' write  ' CAUS &
 1  "<nikajiandikisha>" "andikisha"  V 1/2-SG1-SP VFIN NARR:ka REFL-SG-OBJ SV SVO  ' SVOO  ' write  ' CAUS &
 6  "<kuandikisha>" "andikisha"  V INF  SV SVO  ' SVOO  ' write  ' CAUS &
 3  "<kujiandikisha>" "andikisha"  V INF REFL-SG-OBJ SV SVO  ' SVOO  ' write  ' CAUS &
 1  "<hawajiandikishni>" "andikisha"  V NEG-a VFIN 1/2-PL3-SP REFL-SG-OBJ SV SVO  ' SVOO  ' write  ' CAUS &
 1  "<wakaandikishana>" "andikishana"  V 1/2-PL3-SP VFIN NARR:ka SV SVO  ' SVOO  ' write  ' CAUS REC &
 1  "<wakiandikishwa>" "andikishwa"  V 1/2-PL3-SP VFIN COND:ki SV SVO  ' SVOO  ' write  ' CAUS PASS &
 1  "<waliandikishwa>" "andikishwa"  V 1/2-PL3-SP VFIN PAST  SV SVO  ' SVOO  ' write  ' CAUS PASS &
 1  "<waliandikishwa>" "andikishwa"  V 1/2-PL3-SP VFIN PAST 1/2-PL-REL SV SVO  ' SVOO  ' write  ' CAUS PASS &
 3  "<wameandikishwa>" "andikishwa"  V 1/2-PL3-SP VFIN PERF:me SV SVO  ' SVOO  ' write  ' CAUS PASS &
 1  "<niliandikishwa>" "andikishwa"  V 1/2-SG1-SP VFIN PAST  SV SVO  ' SVOO  ' write  ' CAUS PASS &
 1  "<aliandikishwa>" "andikishwa"  V 1/2-SG3-SP VFIN PAST  SV SVO  ' SVOO  ' write  ' CAUS PASS &
```
Using SALAMA (Swahili Language Manager) in corpus work

1 "<ameandikishwa>" "andikishwa" V 1/2-SG3-SP VFIN PERF:me
SV SVO ' SVOO ' write ' CAUS PASS &
1 "<ziliandikishwa>" "andikishwa" V 9/10-PL-SP VFIN PAST SV
SV ' SVOO ' write ' CAUS PASS &
5 "<kuandikishwa>" "andikishwa" V INF SV SVO ' SVOO ' write
' CAUS PASS &
1 "<hawaandikishwi>" "andikishwa" V NEG-a VFIN 1/2-PL3-SP
SV SVO ' SVOO ' write ' CAUS PASS &
2 "<waandikishwe>" "andikishwa" V SBJN VFIN 1/2-PL2-OBJ SV
SVO ' SVOO ' write ' CAUS PASS "andikishwa" V SBJN VFIN 1/2-PL3-OBJ SV SVO ' SVOO ' write ' CAUS PASS "andikishwa" V SBJN VFIN
1/2-PL3-SP SV SVO ' SVOO ' write ' CAUS PASS &
...