Technical Reports on Language Technology
Report No 36, 20019
http://www.njas.helsinki.fi/salama

Accurate information retrieval using text analysis and disambiguation[1]

Arvi Hurskainen
Department of World Cultures, Box 59
FIN-00014 University of Helsinki, Finland
arvi.hurskainen@helsinki.fi

Abstract

The most common method for retrieving information from text is the string search. In this method, search is targeted to the string in text. This string can consist of a single word, word clusters, partial words etc. Search results are often faulty. The result may have such hits that were not intended. Also, part of targeted words was not retrieved. These problems can be largely overcome by including language analysis into the search system. The report discusses two types of information retrieval implementation, where language analysis and disambiguation are included.

Key Words: *information retrieval, text analysis, constraint grammar.*

1 Introduction

The simple string search method is commonly used in information retrieval. The retreving tool is easy to implement, and the target text needs necessarily no modification. For some languages with little inflection, such as English, this method does the job with some accuracy. Words are usualy found, but if POS information needs to be included, the method fails. English is particularly ambiguous in respect to POS. Many words may be nouns, adjectives and verbs, if only their form is considered. Only the context decides what it is in each case. Therefore, also English produces a lot of unwanted hits when using simple string search.

The situation is still much more complex with languages, which have rich morphology. Words inflect in various ways. Often the inflected sections are as suffixes, but in some language types, such as Bantu languages, verbs inflect in both directions. A verb stem may consist of only a single consonant, and all the rest is inflection. In such cases the plain string search is not possible.

There are search environments for analysed text corpora. In these, the corpus text is analysed and disambiguated and the analysed text is arranged so that each type of tag is placed in a specified column, using the *argument=value* format. Then information can be retrieved from various 'slots'. Such corpus environments allow many types of search possibilities. This is useful especially for research purposes. The normal user usually wants to find particular words in all their surface representations.

[1] The report is issued under licence CC BY-NC

Keeping the needs of non-expert information retrieval in mind, I have developed two retrieval systems. These function in different ways, but they have three things in common. (a) Information retrieval is easy. (b) All occurrences of intended words will be found. (c) No unwanted hits will be retrieved.

2 Analysis of text

The text is first analysed with a suitable analyser. If the aim is to construct a search system, which finds every single word in text, the analyser must be updated using the text as criterion. The analyser must be updated, so that every word in text is analysed correctly. Whether proper names also need be included into the system depends on the language type. For languages such as English and Swahili it is not necessary, because names do not inflect. But in languages such as Finnish, proper names and their inflection codes must be included. The analysed text shows the lemma of each wordform.

3 Disambiguation

Although the lemma of the word is found in analysis, the wordform may have more than one lemma. Also, the POS may be different, although the lemma is the same. Therefore, it is not possible to get precise results, if the analysed words are not disambiguated. There are many types of disambiguation, but here applies the disambiguation of the lemma and its POS tag. No semantic disambiguation or syntactic mapping is needed. For getting maximum accuracy, disambiguation must be developed to high level, using the target text as test material.

4 The format of the analysed text

It is possible to produce analysed and disambiguated text with a varying degree of information. The simplest version is to produce a lemma form of each word, so that the text form and its lemma form are after each other.(1)

(1)
```
1_§ "<*suomi>" "suomi" "<on>" "olla" "<täysivaltainen>"
"täysivaltainen" "<tasavalta>" "tasavalta" "<.$>" "." **CLB
```

A more accurate representation is achieved by adding a POS tag to each lemma (2).

(2)
```
1_§ "<*suomi>" "suomi" N "<on>" "olla" V "<täysivaltainen>"
"täysivaltainen" A "<tasavalta>" "tasavalta" N "<.$>" "." **CLB
```

In my implementations, I follow the latter method, because it gives better results.
Information is usually retrieved with its context. The length of context can be defined in several ways. Sometimes a few words on left and right are sufficient. Also other criteria may be appropriate. In law texts, the paragraph may be the appropriate length. In Bible

texts it is the verse. If there are no specific criteria for defining the context, the sentence is a suitable unit. Very long contexts, however, should be avoided in one of the two types of implementation, because they make the system heavy to implement.

5 Preparing the text for information retrieval

In order to get maximal use of the information retrieval system, the text must be properly processed. As the retrieval system handles individual lines, the unit to be retrieved should be in the form of line. For example, a Bible verse should be handled as a single line. In case of normal prose text, the line could be a sentence. The size of context should be decided in advance, so that the appropriate amoun of context will be retrieved without further measures. Of course, it is possible to shorten context in post-processing phase, but enlarging is more difficult.

6 Information retrieval on the basis of lemma and POS

The information retrieval system, which optimally fulfils the three criteria mentioned above, that is, easy use, full coverage, and accuracy, is the one, which first analyses the key word(s) and then directs the search to the combination of the lemma and its POS tag. It must be noted that the analysis system is used first in processing the text into suitable format, and then in the runtime phase for finding the appropriate combination of lemma and its POS.

When the text is processed, disambiguation can be used for defining precisely the analysis of each word in its context. Whe the analyser is used in the runtime phase, it is understandable that disambiguation cannot be used, because there is no context. This problem can be solved often by entering such wordform into the search box, which would not produce ambiguous readings. Note that the system analyses any form of the word, including all kinds of clitics.

The construction of this retrieval system includes several phases. After analysis and disambiguation, the text has a format such as in (3).

(3)
```
1_§_2 "<*suomen>" "suomi" PROPN "<valtiosääntö>" "valtiosääntö" N
"<on>" "olla" V "<vahvistettu>" "vahvistaa" V "<tässä>" "tämä"
PRON "<perustuslaissa>" "perustuslaki" N "<.$>" "." **CLB
```

The identification code *1_§_2* initiates each line. Then follows each surface wordform and its lemma plus POS. All the rest from the analysis is removed. In the next phase we modify the sentence to be more suitable for retrieval (4).

(4)
```
1_§_2 "<*suomen>" [suomi_PROPN] "<valtiosääntö>" [valtiosääntö_N]
"<on>" "olla" V "<vahvistettu>" [vahvistaa_V] "<tässä>" "tämä"
PRON "<perustuslaissa>" [perustuslaki_N] "<.$>" "." **CLB
```

Note that some very common words, such as *on* and *tässä* do not have the lemma plus POS combination. This is done for increasing processing speed. Hardly anybody would search those words anyway. Then we mark each lemma with Roman numbers. Using Arabic numbers would cause problems with Perl code (5). Note that also the line number '3.' is added.

(5)
```
3. 1_§_2 "<*suomen>" I[suomi_PROPN] "<valtiosääntö>"
II[valtiosääntö_N] "<on>" "olla" V "<vahvistettu>"
III[vahvistaa_V] "<tässä>" "tämä" PRON "<perustuslaissa>"
IV[perustuslaki_N] "<.$>" "." **CLB
```

Then we make a copy of the sentence for each lemma, which is supposed to be retrievable in the system. All other lemmas are removed (6).

(6)
```
[suomi_PROPN]] 3. 1_§_2 Suomen [suomi_PROPN] valtiosääntö on
  vahvistettu tässä perustuslaissa.
[valtiosääntö_N]] 3. 1_§_2 Suomen valtiosääntö [valtiosääntö_N] on
  vahvistettu tässä perustuslaissa.
[vahvistaa_V]] 3. 1_§_2 Suomen valtiosääntö on vahvistettu
  [vahvistaa_V] tässä perustuslaissa.
[perustuslaki_N]] 3. 1_§_2 Suomen valtiosääntö on vahvistettu
  tässä perustuslaissa [perustuslaki_N].
```

Now we have a unique sentence for each lemma. The lemma is also copied to the beginning of the line. Then we convert these lines into Beta rules, such as in (7).

(7)
```
suomi_N; 3. 1_§_2 Suomen [suomi_PROPN] valtiosääntö on vahvistettu
  tässä perustuslaissa.;
valtiosääntö_N; 3. 1_§_2 Suomen valtiosääntö [valtiosääntö_N] on
  vahvistettu tässä perustuslaissa.;
vahvistaa_V; 3. 1_§_2 Suomen valtiosääntö on vahvistettu
  [vahvistaa_V] tässä perustuslaissa.;
perustuslaki_N; 3. 1_§_2 Suomen valtiosääntö on vahvistettu tässä
  perustuslaissa [perustuslaki_N].;
```

When the Beta rules are applied, the retrieval result has the form as in (8).

(8)
```
3. 1_§_2 Suomen [suomi_PROPN] valtiosääntö on vahvistettu tässä
  perustuslaissa.
3. 1_§_2 Suomen valtiosääntö [valtiosääntö_N] on vahvistettu tässä
  perustuslaissa.
3. 1_§_2 Suomen valtiosääntö on vahvistettu [vahvistaa_V] tässä
  perustuslaissa.
3. 1_§_2 Suomen valtiosääntö on vahvistettu tässä perustuslaissa
  [perustuslaki_N].
```

Line numbering is useful in case of large texts, which should be divided into several rule files for ensuring smooth processing. The numbers help to reorder the retrieved lines into desired order.

The retrieval system can be used in two ways. It is possible to enter the combination of the lemma and POS, such as *{perustuslaki_N}* into the search box (or with echo on the command line). The beta rules expect that the left boundary is '{' and the right boundary is '}'.

The other, and much more convenient, way is to use the analyser to find the correct form for retrieval. In this method, you can enter any form of the word, and the analyser finds its default lemma and POS.

The specific boundary marks are important in that they help in defining the lemma precisely. For example, if boundary marks are excluded, by entering *laki_N* the system would find *[laki_N]* and *[perustuslaki_N].*

If you enter any form of the word *laki*, such as *laki*, *lain*, *lailla*, *laissa*, *laissakin*, *laillakaan* etc., you would get the analysis result *{laki_N}*. This would match with the corresponding Beta rule and all lines with this lemma plus POS would be retrieved. Note that *[perustuslaki_N]* would be excluded, because the left boundary mark '{' prevents it.

Also note that the lemma always starts with a lowercase letter, also in proper names. This arrangement helps in formulating the search string, becuse one does not need to bother about capital letters. Only in the final phase, proper names are converted into capital-initial form.

7 Information retrieval on the basis of surface text or lemma

There is also another method to implement a precise information retrieval system. The text is first analysed and disambiguated, and the analysed text is converted into the format as in (4) above. Then it can be cleaned a bit, so that we get the form as in (9).

(9)
```
1_§_2 Suomen [suomi_PROPN] valtiosääntö [valtiosääntö_N] on
  vahvistettu [vahvistaa_V] tässä perustuslaissa
  [perustuslaki_N].
```

Search can now be directed to this text format. If we use the search applications of the *grep* family, widely used in Linux and other environments, we can direct the search to any string in text. If the search is targeted to surface text, the search system behaves as the normal string search. For example, if the search string is *perustusla*, the system finds lines, which have the strings *perustuslaissa*, *perustuslaillinen*, *perustuslaillisessa*, *perustuslakivaliokunnan*, *perustuslakivaliokunnassa* etc.

If we use the search string *perustuslaki_N*, the system finds only such words, which have the base form *perustuslaki_N*. Even the search string *perustuslaki_* would find the same hits, because the word can be only a noun.

On the other hand, if we use the search string *laki_N*, we would get occurrences of *[laki_N]* and *[perustuslaki_N]*. This can be prevented by using the left boundary mark, which is here '['. Thus the search string *[laki_* would find only the hits, which have *[laki_N]* as base form.

In this second system, the user sees only the final result, where all lemmas and POS tags have been removed. The hit can be emphasised with color if needed.

The advantages of this search system include its varying search possiblities. Also its implementation is much easier than of the first method, because a single analysed text file serves as source text. The setback is that the hit cannot be pointed out as precisely as in the first system. Also its implementation using runtime analysis has not so far succeeded, deriving from different string demarkating criteria in Beta and *egrep*.

8 Including multiword expressions

It is also possible to search for multiword expressions from text. Consider examples from the search system of the Swahili Bible (10).

(10)
```
{ona_njaa_V}
Yer_42.14 mkisema, La lakini tutakwenda nchi ya Misri, ambayo
hatutaona vita, wala kuisikia sauti ya tarumbeta, wala kuona njaa
[ona_njaa_V] kukosa chakula; nasi tutakaa huko;
Yoh_6.35 Yesu akawaambia, Mimi ndimi chakula cha uzima; yeye ajaye
kwangu hataona njaa [ona_njaa_V] kabisa, naye aniaminiye hataona
kiu kamwe.
Fun_7.16 Hawataona njaa [ona_njaa_V] tena, wala hawataona kiu
tena, wala jua halitawapiga, wala hari iliyo yote.
```

The Swahili Bible has three occurrences of the MWE *ona_njaa_V*, ‘to be hungry’, literally 'to see hunger’. The combination of a verb *ona* and noun *njaa* consists of a MWE, which has a function of a verb. The MWEs are displayed so that each member of the cluster is represented by its lemma in the cluster, separated by underscore '_'. The MWE lemma is located immediately after the surface cluster of the MWE. We see in (10) that the lemma *ona_njaa_V* has the surface forms *kuona njaa*, *hataona njaa*, and *Hawataona njaa* in text.

We can make another test and search for the MWE *ona_kiu_V*, 'to be thirsty' (11).

(11)
```
Amu_15.18 Kisha akaona kiu [ona_kiu_V] sana, akamwita Bwana
akasema, Wewe umetupa wokovu huu kwa mkono wa mtumishi wako; na
sasa nitakufa kwa kiu, na kuanguka katika mikono ya watu
wasiotahiriwa.
Isa_48.21 Wala hawakuona kiu [ona_kiu_V] alipowaongoza jangwani;
Yoh_4.13 Yesu akajibu, akamwambia, Kila anywaye maji haya ataona
kiu [ona_kiu_V] tena;
Yoh_4.14 walakini ye yote atakayekunywa maji yale nitakayompa mimi
hataona kiu [ona_kiu_V] milele; bali yale maji nitakayompa
yatakuwa ndani yake chemchemi ya maji, yakibubujikia uzima wa
milele.
Yoh_6.35 Yesu akawaambia, Mimi ndimi chakula cha uzima; yeye ajaye
kwangu hataona njaa kabisa, naye aniaminiye hataona kiu
[ona_kiu_V] kamwe.
```

```
Fun_7.16 Hawataona njaa tena, wala hawataona kiu [ona_kiu_V] tena,
wala jua halitawapiga, wala hari iliyo yote.
```

These six hits found from the Bible show that one of them was used in affirmative meaning and five in negative meaning.
Note that if one is searching MWEs using the first search system, one may use any form of the verb, for example, *aliona kiu*, *walioona kiu*, *watakaoona kiu*, *wasipoona kiu*, *asingeona kiu*, *wasingaliona kiu*, or any other of the thousands of verb forms.

9 Conclusion

The report described two methods for implementing advanced information retrieval. Both make use of text analysis and disambiguation, so that search can be targeted to the lemma and its accompanying POS tag instead of surface text. This method makes the search comprehensive and precise. In one method, also the possibility to use runtime text analysis for converting the surface word into desired format makes the search easy and convenient for the user.