Lab 4 - Extraction of subject-verb-object triples

Laboration 4 in EDAN20 @ LTH - [http://cs.lth.se/edan20/coursework/assignment-4/ (http://cs.lth.se/edan20/coursework/assignment-4/)](http://cs.lth.se/edan20/coursework/assignment-4/)

Author: Jonatan Kronander

Objectives

The objectives of this assignment are to:

- Extract the subject–verb pairs from a parsed corpus
- Extend the extraction to subject–verb–object triples
- Understand how dependency parsing can help create a knowledge base
- Write a short report of 1 to 2 pages on the assignment

This assignment is inspired by the Prismatic knowledge base used in the IBM Watson system. See this paper for details.

In this session, you will first use a parsed corpus of Swedish to extract the pairs and triples, and then apply it to other languages.

Choosing a parsed corpus

In this part, you will use the CONLL-X Swedish corpus. Download the tar archives containing the training and test sets for Swedish and uncompress them: [data sets].

In [1]:
```python
from urllib.request import urlopen

b_train_text = urlopen("http://fileadmin.cs.lth.se/cs/Education/EDAN20/corpus/conllx/sv/swedish_talbanken05_train.conll").read() # Open file and read
train_text = str(b_train_text,'utf-8')

b_test_text = urlopen("http://fileadmin.cs.lth.se/cs/Education/EDAN20/corpus/conllx/sv/swedish_talbanken05_test_blind.conll").read() # Open file and read
test_text = str(b_test_text,'utf-8')
```

In [2]:
```python
f_out = open('train_sen.cnoll', 'w')

for sentence in train_text:
    f_out.write(sentence)
f_out.close()
```

In [3]:
```python
print(train_text[:550])
```

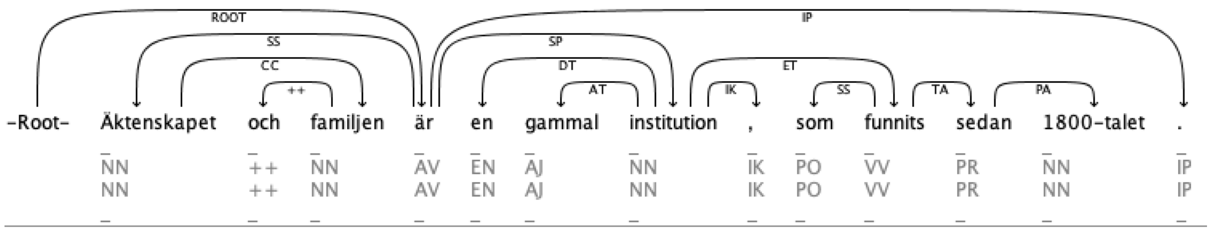
```
1       Äktenskapet     _       NN      NN      _       4       SS      _       _
2       och     _       ++      ++      _       3       ++      _       _
3       familjen        _       NN      NN      _       1       CC      _       _
4       är      _       AV      AV      _       0       ROOT    _       _
5       en      _       EN      EN      _       7       DT      _       _
6       gammal  _       AJ      AJ      _       7       AT      _       _
7       institution     _       NN      NN      _       4       SP      _       _
8       ,       _       IK      IK      _       7       IK      _       _
9       som     _       PO      PO      _       10      SS      _       _
10      funnits _       VV      VV      _       7       ET      _       _
11      sedan   _       PR      PR      _       10      TA      _       _
12      1800-talet      _       NN      NN      _       11      PA      _       _
13      .       _       IP      IP      _       4       IP      _       _

1       Är      _       AV      AV      _       0       ROOT    _       _
2       den     _       PO      PO      _       1       SS      _       _
3       berättigad      _       TP      TP      _       1       SP      _       _
4       i       _       PR      PR      _       1       RA      _       _
5       dagens  _       NN      NN      _       6       DT      _       _
6       samhälle        _       NN      NN      _       4       PA      _       _
7       .       _       IP      IP      _       1       IP      _
```

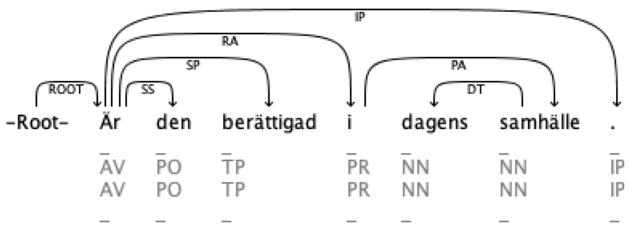
Draw a graphical representation of the two first sentences of the training set.

Download What's wrong with my NLP and use it to check your representations.

First sentence: Äktenskapet och familjen är en gammal institution, som funnits sedan 1800-talet.

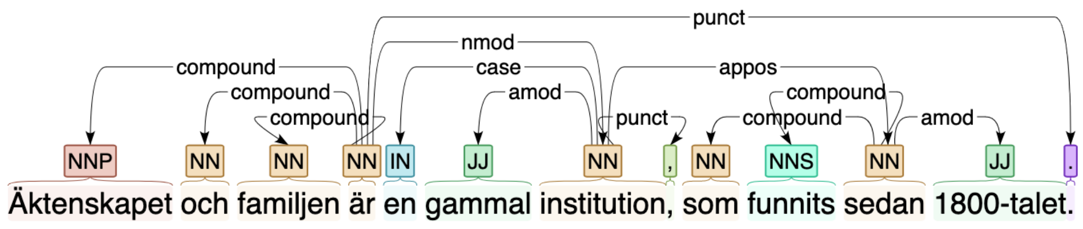

ROOT
SS
CC
++
SP
DT
AT
IP
ET
IK
SS
TA
PA
-Root- Äktenskapet och familjen är en gammal institution , som funnits sedan 1800-talet .
NN ++ NN AV EN AJ NN IK PO VV PR NN IP

Second sentence: Är den berättigad i dagens samälle.

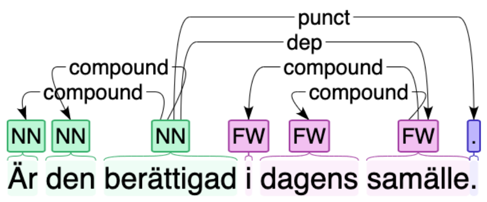


Apply the dependency parser for Swedish of the Langforia pipelines to these sentences. Link to Lanforia pipelines: [http://vilde.cs.lth.se:9000/ (http://vilde.cs.lth.se:9000/)](http://vilde.cs.lth.se:9000/)

First sentence:



Second sentence:



Extracting the subject–verb pairs

Extract all the subject–verb pairs and the subject–verb–object triples from the training corpus. First get corpus. Based on course code found here: https://github.com/pnugues/ilppp/blob/master/programs/labs/relation_extraction/python/conll.py (https://github.com/pnugues/ilppp/blob/master/programs/labs/relation_extraction/python/conll.py)

Conll tag decription:

Field number:	Field name:	Description:
1	ID	Token counter, starting at 1 for each new sentence.
2	FORM	Word form or punctuation symbol.
3	LEMMA	Lemma or stem (depending on particular data set) of word form, or an underscore if not available.
4	CPOSTAG	Coarse-grained part-of-speech tag, where tagset depends on the language.
5	POSTAG	Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available.
6	FEATS	Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (|), or an underscore if not available.
7	HEAD	Head of the current token, which is either a value of ID or zero ('0'). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero.
8	DEPREL	Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningfull or simply 'ROOT'.
9	PHEAD	Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. Note that depending on the original treebank annotation, there may be multiple tokens an with ID of zero. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all languages), whereas the structures resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).
10	PDEPREL	Dependency relation to the PHEAD, or an underscore if not available. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningfull or simply 'ROOT'.

In [4]:
```
column_names = ['id', 'form', 'lemma', 'cpostag', 'postag', 'feats', 'head', 'deprel', 'phead', 'pdeprel']

#sentences = read_sentences(train_file)
train_text = train_text.strip()
train_sentences = train_text.split('\n\n')
```

Creates a list of sentence where each sentence is a list of lines, each line is a dictionary of columns.

In [5]:
```
#formatted_corpus = split_rows(sentences, column_names_2006)
sentences = []
root_values = ['0', 'ROOT', 'ROOT', 'ROOT', 'ROOT', 'ROOT', '0', 'ROOT', '0', 'ROOT']
start = [dict(zip(column_names, root_values))]

for sentence in train_sentences:
    rows = sentence.split('\n')
    sentence = [dict(zip(column_names, row.split('\t'))) for row in rows if row[0] != '#']
    sentence = start + sentence
    sentences.append(sentence)

print(sentences[0][1]) #sentences[mening][ord]
```

```
{'id': '1', 'form': 'Äktenskapet', 'lemma': '_', 'cpostag': 'NN', 'postag': 'NN', 'feats': '_', 'head': '4', 'deprel': 'SS', 'phead': '_', 'pdeprel': '_'}
```

```
In [6]: def nsubj_pairs(sentences):
            """
            Returns all subject-verb pairs from ConllX format list
            :param list:
            :return: dict of all subject-verb pairs (keys = tuple)
            """
            nsubj_dict = {}
            for sent in sentences: # Iterate all sentences
                for row in sent: # Iterate each sentence
                    deprel_row = row['deprel']
                    if deprel_row == 'SS': # If word is subject
                        SS_word_index = row['head']
                        SS_word = sent[int(SS_word_index)]['form'].lower()
                        NN_word = row['form'].lower()
                        if (NN_word,SS_word) in nsubj_dict: # Add to dict
                            nsubj_dict[(NN_word, SS_word)] += 1
                        else:
                            nsubj_dict[(NN_word,SS_word)] = 1
            return nsubj_dict
```

```
In [7]: nsubj_dict = nsubj_pairs(sentences)
```

```
In [8]: print("Total number of subject-verb pairs found: " + str(sum(list(nsubj_dict.values()))))
```

```
Total number of subject-verb pairs found: 18885
```

```
In [9]: import operator
        print("The most frequent subject-verbs are: ")

        sorted_nsubj_list = sorted(nsubj_dict.items(), key=operator.itemgetter(1))
        top_5 = sorted_nsubj_list[len(sorted_nsubj_list)-5:]

        for pair in reversed(top_5):
            print(pair[0][0] +"-"+ pair[0][1] + ": " + str(pair[1]))
```

```
The most frequent subject-verbs are:
det-är: 537
man-kan: 261
som-är: 211
jag-tror: 171
äktenskapet-är: 161
```

Extracting the subject–verb–object triples

Using same training corpus as above.

```
In [10]: def nsubjo_triples(sentences):
             """
             Returns all subject-verb pairs from ConllX format list
             :param list:
             :return: dict of all subject-verb pairs (keys = tuple)
             """
             nsubjo_dict = {}
             for sent in sentences: # Iterate all sentences
                 for row in sent: # Iterate each sentence
                     deprel_row = row['deprel']
                     if deprel_row == 'SS': # If word is subject
                         NN_word = row['form'].lower()
                         SS_word_index = row['head']
                         SS_word = sent[int(SS_word_index)]['form'].lower()
                         for row_2 in sent: # Iterate sentence again and look for object
                             deprel_row_2 = row_2['deprel']
                             if deprel_row_2 == 'OO':
                                 OO_word = row_2['form'].lower()
                                 if row_2['head'] == row['head']: # Is subject and object pointing at same verb?
                                     if (NN_word,SS_word,OO_word) in nsubjo_dict: # Add to dict
                                         nsubjo_dict[(NN_word,SS_word,OO_word)] += 1
                                     else:
                                         nsubjo_dict[(NN_word,SS_word,OO_word)] = 1
             return nsubjo_dict
```

```
In [11]: nsubjobj_dict = nsubjo_triples(sentences)
```

```
In [12]: print("Total number of subject-verb-object triples found: " + str(sum(list(nsubjobj_dict.values()))))
```

```
Total number of subject-verb-object triples found: 5844
```

```
In [13]: print("Most common subject-verb-object triples found in Swedish corpus:")

         sorted_nsubjo_list = sorted(nsubjobj_dict.items(), key=operator.itemgetter(1))
         top_5 = sorted_nsubjo_list[len(sorted_nsubjo_list)-5:]

         for pair in reversed(top_5):
             print(pair[0][0] +"-"+ pair[0][1] + "-" + pair[0][2] +": " + str(pair[1]))
```

```
Most common subject-verb-object triples found in Swedish corpus:
man-gifter-sig: 37
jag-tycker-är: 36
jag-tror-är: 36
man-vänder-sig: 19
som-ingått-äktenskap: 17
```

Multilingual Corpora

Applying the model it to all the other languages from Universal Dependencies repository. https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2988 (https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2988)

Read CoNLL-U format:

```
In [14]: swedish_path = "corps/ud-treebanks-v2.4/UD_Swedish-LinES/sv_lines-ud-train.conllu" #Right

         swedish_text = open(swedish_path).read()
```

```
In [15]: print(swedish_text[0:500])
```

```
# newdoc
# sent_id = 1
# text = Visa alla
1       Visa    visa    VERB    IMP-ACT Mood=Imp|VerbForm=Fin|Voice=Act 0       root    _       _
2       alla    all     PRON    TOT-PL-NOM      Definite=Ind|Number=Plur|PronType=Tot   1       obj     _       _

# sent_id = 2
# text = Om ANSI SQL-frågeläge
1       Om      om      ADP     _       _       3       case    _       _
2       ANSI    ANSI    PROPN   SG-NOM  Case=Nom        3       nmod    _       _
3       SQL-frågeläge   SQLfrågeläge    NOUN    IND-NOM Case=Nom|Definite=Ind|Gender=Neut|Number=Sing   0       root    _       _

# sent_id = 3
# text = En del av innehållet i det här avsnittet kanske inte gäller för vissa språ
```

```
In [16]: def txt_to_conllu(text, column_names = ['id', 'form', 'lemma', 'upos', 'xpos', 'feats', 'head', 'deprel', 'deps', 'misc'], root_values = ['0', 'ROOT', 'ROOT', 'ROOT', 'ROOT', 'ROOT', '0', 'ROOT',
         '0', 'ROOT']):
             text = text.strip()
             text_sentences = text.split('\n\n')

             sentences = []
             start = [dict(zip(column_names, root_values))]
             for sentence in text_sentences:
                 rows = sentence.split('\n')
                 sentence = [dict(zip(column_names, row.split('\t'))) for row in rows if row[0] != '#']
                 sentence = start + sentence

                 for word_dict in sentence: # Removing "multiwords" from corpus
                     if '-' in word_dict['id']:
                         sentence.remove(word_dict)

                 sentences.append(sentence)

             return sentences
```

In [17]:
```
swedish_corpus_dict = txt_to_conllu(swedish_text, column_names)

#swedish_corpus_dict[:5][:] #sentences[mening][ord]
```

In [18]:
```
def subjverb_pairs(sentences):
    """
    Returns all subject-verb pairs from ConllU format list
    :param list:
    :return: dict of all subject-verb pairs (keys = tuple)
    """
    subjverb_dict = {}
    for sent in sentences: # Iterate all sentences
        for row in sent: # Iterate each sentence
            deprel_row = row['deprel']
            if deprel_row == 'nsubj': # If word is subject
                subj_word_index = row['head']
                subj_word = sent[int(subj_word_index)]['form'].lower()
                verb_word = row['form'].lower()
                if (verb_word,subj_word) in subjverb_dict: # Add to dict
                    subjverb_dict[(verb_word,subj_word)] += 1
                else:
                    subjverb_dict[(verb_word,subj_word)] = 1

    return subjverb_dict
```

In [19]:
```
swedish_subjverb = subjverb_pairs(swedish_corpus_dict)
```

In [20]:
```
print("Total number of subject-verb pairs found in Swedish corpus: " + str(sum(list(swedish_subjverb.values()))))
```
```
Total number of subject-verb pairs found in Swedish corpus: 4469
```

In [21]:
```
print("The most frequent subject-verbs, in Swedish corpus, are: ")

swedish_sorted_subjverb_list = sorted(swedish_subjverb.items(), key=operator.itemgetter(1))
top_5 = swedish_sorted_subjverb_list[len(swedish_sorted_subjverb_list)-5:]

for pair in reversed(top_5):
    print(pair[0][0] +"-"+ pair[0][1] + ": " + str(pair[1]))
```
```
The most frequent subject-verbs, in Swedish corpus, are:
han-hade: 24
han-sa: 22
han-gick: 22
jag-vet: 18
han-såg: 17
```

In [22]:
```
def subjverbobj_triples(sentences):
    """
    Returns all subject-verb pairs from ConllX format list
    :param list:
    :return: dict of all subject-verb pairs (keys = tuple)
    """
    subjverbobj_dict = {}
    for sent in sentences: # Iterate all sentences
        for row in sent: # Iterate each sentence
            deprel_row = row['deprel']
            if deprel_row == 'nsubj': # If word is subject
                subj_word = row['form'].lower()
                verb_word_index = row['head']
                verb_word = sent[int(verb_word_index)]['form'].lower()
                for row_2 in sent: # Iterate sentence again and look for object
                    deprel_row_2 = row_2['deprel']
                    if deprel_row_2 == 'obj':
                        obj_word = row_2['form'].lower()
                        if row_2['head'] == row['head']: # Is subject and object pointing at same verb?
                            if (subj_word,verb_word,obj_word) in subjverbobj_dict: # Add to dict
                                subjverbobj_dict[(subj_word,verb_word,obj_word)] += 1
                            else:
                                subjverbobj_dict[(subj_word,verb_word,obj_word)] = 1
    return subjverbobj_dict
```

In [23]:
```
swedish_subjverbobj = subjverbobj_triples(swedish_corpus_dict)
```

In [24]:
```
print("Total number of subject-verb-object triples found in Swedish corpus: " + str(sum(list(swedish_subjverbobj.values()))))
```
```
Total number of subject-verb-object triples found in Swedish corpus: 1733
```

In [25]:
```
print("Most common subject-verb-object triples found in Swedish corpus:")

swedish_sorted_subjverbobj_list = sorted(swedish_subjverbobj.items(), key=operator.itemgetter(1))
top_5 = swedish_sorted_subjverbobj_list[len(swedish_sorted_subjverbobj_list)-5:]

for pair in reversed(top_5):
    print(pair[0][0] +"-"+ pair[0][1] + "-" + pair[0][2] +": " + str(pair[1]))
```
```
Most common subject-verb-object triples found in Swedish corpus:
han-befann-sig: 6
jag-visste-det: 4
han-beslöt-sig: 4
han-sa-sig: 4
du-flyttar-linje: 4
```

Lets try it on other languages. For example Slovak and English:

In [26]:
```
slovak_path = "corps/ud-treebanks-v2.4/UD_Slovak-SNK/sk_snk-ud-train.conllu"
slovak_text = open(slovak_path).read()

slovak_corpus_dict = txt_to_conllu(slovak_text)
slovak_subjobj = subjverb_pairs(slovak_corpus_dict)
slovak_subjverbobj = subjverbobj_triples(slovak_corpus_dict)

print("----------------------")
print("Total number of subject-verb pairs found in Slovak corpus: " + str(sum(list(slovak_subjobj.values()))))
print("----------------------")
print("The most frequent subject-verbs, in Slovak corpus, are: ")

slovak_sorted_subjobj_list = sorted(slovak_subjobj.items(), key=operator.itemgetter(1))
top_5 = slovak_sorted_subjobj_list[len(slovak_sorted_subjobj_list)-5:]

for pair in reversed(top_5):
    print(pair[0][0] +"-"+ pair[0][1] + ": " + str(pair[1]))
print("----------------------")
print("Total number of subject-verb-object triples found in Slovak corpus: " + str(sum(list(slovak_subjverbobj.values()))))
print("----------------------")
print("Most common subject-verb-object triples found in Slovak corpus:")

slovak_sorted_subjverbobj_list = sorted(slovak_subjverbobj.items(), key=operator.itemgetter(1))
top_5 = slovak_sorted_subjverbobj_list[len(slovak_sorted_subjverbobj_list)-5:]

for pair in reversed(top_5):
    print(pair[0][0] +"-"+ pair[0][1] + "-" + pair[0][2] +": " + str(pair[1]))
```
```
----------------------
Total number of subject-verb pairs found in Slovak corpus: 5480
----------------------
The most frequent subject-verbs, in Slovak corpus, are:
vláda-pripraví: 16
to-je: 13
vláda-podporovať: 10
chris-povedal: 10
chris-odvetil: 10
----------------------
Total number of subject-verb-object triples found in Slovak corpus: 1891
----------------------
Most common subject-verb-object triples found in Slovak corpus:
vláda-venovať-pozornosť: 6
vláda-vytvorí-podmienky: 4
srdce-poskočilo-mu: 4
abu-pokrčil-plecami: 3
tvár-zjavila-mi: 3
```

In [27]:
```
english_path = "corps/ud-treebanks-v2.4/UD_English-EWT/en_ewt-ud-train.conllu"
english_text = open(english_path).read()
english_corpus_dict = txt_to_conllu(english_text)

english_subjobj = subjverb_pairs(english_corpus_dict)

print("----------------------")
print("Total number of subject-verb pairs found in English corpus: " + str(sum(list(slovak_subjobj.values()))))
print("----------------------")
print("The most frequent subject-verbs, in English corpus, are: ")

english_sorted_subjobj_list = sorted(english_subjobj.items(), key=operator.itemgetter(1))
top_5 = english_sorted_subjobj_list[len(english_sorted_subjobj_list)-5:]

for pair in reversed(top_5):
    print(pair[0][0] +"-"+ pair[0][1] + ": " + str(pair[1]))
print("----------------------")
print("Total number of subject-verb-object triples found in English corpus: " + str(sum(list(english_subjobj.values()))))
print("----------------------")
print("Most common subject-verb-object triples found in English corpus:")

english_subjverbobj = subjverbobj_triples(english_corpus_dict)

english_sorted_subjverbobj_list = sorted(english_subjverbobj.items(), key=operator.itemgetter(1))
top_5 = english_sorted_subjverbobj_list[len(english_sorted_subjverbobj_list)-5:]

for pair in reversed(top_5):
    print(pair[0][0] +"-"+ pair[0][1] + "-" + pair[0][2] +": " + str(pair[1]))
```

```
----------------------
Total number of subject-verb pairs found in English corpus: 5480
----------------------
The most frequent subject-verbs, in English corpus, are:
you-have: 188
i-have: 174
i-had: 117
i-think: 105
i-know: 98
----------------------
Total number of subject-verb-object triples found in English corpus: 16258
----------------------
Most common subject-verb-object triples found in English corpus:
you-have-questions: 22
you-think-what: 12
i-do-what: 7
i-have-idea: 6
i-do-it: 6
```

Reading

Read the article: PRISMATIC: Inducing Knowledge from a Large Scale Lexicalized Relation Resource by Fan and al. (2010) and write in a few sentences how it relates to your work in this assignment. https://www.aclweb.org/anthology/W10-0915 (https://www.aclweb.org/anthology/W10-0915)

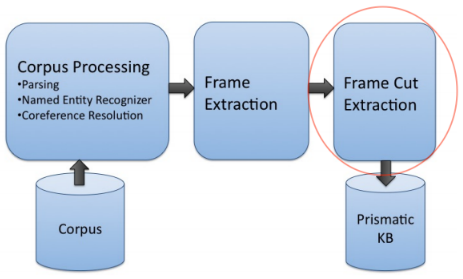


Figure 1: System Overview

Figure 1, from paper above, is a pipeline of PRISMATIC and in this lab we have done some parts of the step "Frame Cut Extraction". Read more under section 6 "Frame cut" in Fan and al. paper (2010). The "frame cut extraction" in Prismatic seems to extract more frame cuts, e.g. S-V-O-IO, S-V-P-O (where S - subject, V - verb, O - object, IO - indirect object). Prismatic does all corpus processing and parsing whereas we take already annotated corpus, "Universal Dependencies".