Language Technology - EDAN20
Assignment 5 - Fall 2020

Extraction of Subject-Verb-Object Triples
CoNLL-U Annotation

Hicham Mohamad, hi8826mo-s

November 13, 2020

1 Introduction

In this assignment, we are going to practice and deal with **relation extraction**, one of infromation extraction (IE) tasks, and understand how **dependency parsing** can help create a knowledge base. We will extract all the subject–verb pairs and subject–verb–object triples from a parsed corpus involving two words or entities. The implemented program will use a parsed corpus of Swedish to extract the pairs and triples and then it will be applied to other languages. This work is inspired by the **Prismatic knowledge base** used in the IBM Watson system [9].

2 UD Parsed corpus - CoNLL-U annotation

By using the CoNLL-U reader program, we read/download the latest version of Universal Dependencies (UD 2.6) [6], which contains **treebanks** for 60+ languages. Then, we need to extract all the subject–verb pairs and the subject–verb–object triples from the **Swedish Talbanken** training corpus. Annotations in CoNLL-U format are encoded in plain text files where the word lines contain the annotation of a word/token in 10 fields separated by single tab characters. The 10 column names of the CoNLL-U corpora consists of

```
['ID', 'FORM', 'LEMMA', 'UPOS', 'XPOS', 'FEATS', 'HEAD', 'DEPREL', 'DEPS', 'MISC']
```

Graphical representations

In the task, we need to carry out different visualizations for the two first Swedish sentences of the training set of the Swedish Talbanken corpus, using the CoNLL-U annotation. These first sentences are:

1. Individuell beskattning av arbetsinkomster
2. Genom skattereformen införs individuell beskattning (särbeskattning) av arbetsinkomster.

Drawings In this first step, we obtain the drawings illustrated in Figure 1

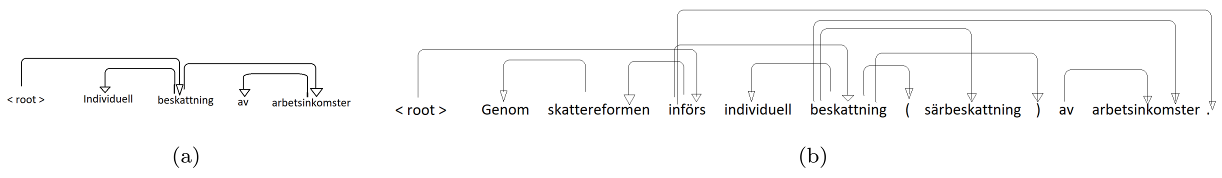
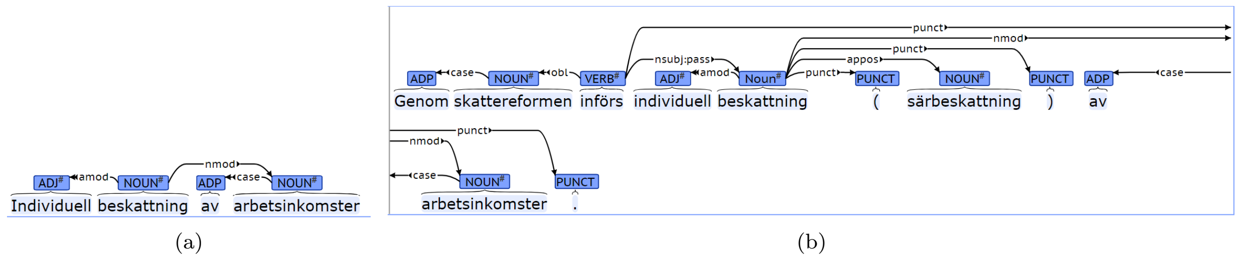


Figure 1: Drawing the dependency graph to the two first Swedish sentences of the training set. a) Sentence 1. b) Sentence 2.

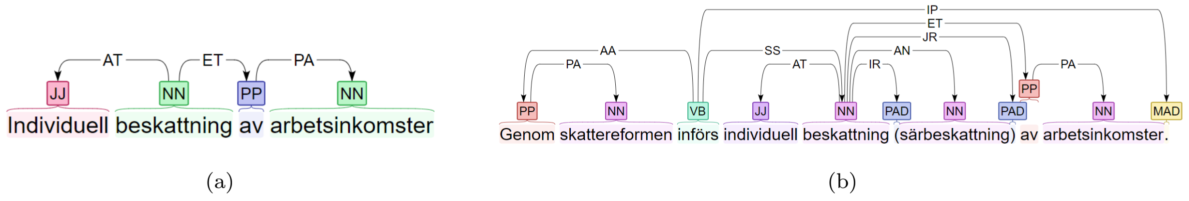
Editable visualization conllu.js Here using CoNLL-U format library for JavaScript, we can visualize these sentences with this tool: `http://spyysalo.github.io/conllu.js/`, the resulting graphical representations are shown in Figure 2



ADJ NOUN ADP NOUN
amod nmod case
Individuell beskattning av arbetsinkomster
(a)
ADP NOUN VERB ADJ Noun PUNCT NOUN PUNCT ADP
case obl nsubj:pass amod punct appos punct nmod case
Genom skattereformen införs individuell beskattning (särbeskattning) av
nmod punct case
NOUN PUNCT
arbetsinkomster .
(b)

Figure 2: Applying the editable visualization (conllu.js) generated from CoNLL-U formatted input to the two first Swedish sentences of the training set. a) Sentence 1. b) Sentence 2.

Language pipelines In this part, we apply the dependency parser for Swedish of the **Langforia pipelines** to these sentences. On the web site to Lanforia pipelines: `http://vilde.cs.lth.se:9000/`, we have to select `Swedish` and activate both `Token` and `DependencyRelation`.



AT ET PA
JJ NN PP NN
Individuell beskattning av arbetsinkomster
(a)
IP ET JR AA PA SS AT AN IR PA
PP NN VB JJ NN PAD NN PAD PP NN MAD
Genom skattereformen införs individuell beskattning (särbeskattning) av arbetsinkomster.
(b)

Figure 3: Applying the dependency parser for Swedish of the Langforia pipelines to the two first Swedish sentences of the training set. a) Sentence 1. b) Sentence 2.

3 Extracting pairs and triples

In this section, we need to extract all the **subject–verb** pairs and the **subject–verb–object** triples from the Swedish Talbanken training corpus. The implemented program is designed following these steps:

- Reading the training corpus: To start the program, we use the provided **CoNLL-U reader** which works also to read the other corpora. The resulting formatted corpus contains 4303 sentences, where each sentence is a list of lines and each line is a dictionary of columns.
- Converting the lists in Python's dictionaries: To ease the processing of some corpora, we need to use a **dictionary represention** of the sentences. The keys will be the ID values. We do this because ID is not necessarily a number.
- Extracting all the subject-verb pairs in the corpus: First we need to set the words in lowercase, and we implement the function `extract_pairs(formatted_corpus_dict)` where:
  - In the extraction, we just check the function between two words.
  - We return the results as Python's dictionaries, where the key is the pair as tuples and the value is the count. The computed total number of pairs is 6,083 subject-verb pairs.
- Finding the most frequent pairs: Here we sort the pairs by frequency and by lexical order, and then we get the three most frequent pairs. Here are the resulting most frequent pairs we find:

  ```
  [(('som', 'har'), 45), (('du', 'får'), 19), (('vi', 'har'), 19)]
  ```

- Extracting all the subject–verb–object triples of the corpus: we implement the function `extract_triples()` where the object function uses the `obj` code. Then the computed total number of triples is 2054 subject-verb-object triples.

- Finding the most frequent triples: We sort the obtained triples by frequency and by lexical order and get the three most frequent triples. Here are the most frequent triples we find:

  ```
  [(('man', 'vänder', 'sig'), 14),
  (('det', 'rör', 'sig'), 5),
  (('man', 'söker', 'arbete'), 3)]
  ```

4 Multilingual Corpora

Now, after making it work on Swedish training corpus, we need to apply the implemented program to all the other languages in the repository: **Universal Dependencies**. The provided function `get_files()` returns all the files from a folder with a suffix. As in the previous sections above, we consider the training files only. Here are the principal points of the implemented program to carry out this task:

1. Dealing with the indices: As shown in Table 1 below some corpora expand some tokens into multiwords, as the case in French, Spanish, and German.

French	Spanish	German
du: de le	*del*: de el	*zur*: zu der
des: de les	*vámonos*: vamos nos	*im*: in dem

Table 1: Examples of some expansions.

   As described in details in **CoNLL-U format**, in the corpora we have the original tokens as well as the multiwords as for example in *vámonos al mar* we have:

   ```
   1-2 vámonos _
   1 vamos ir
   2 nos nosotros
   3-4 al _
   3 a a
   4 el el
   5 mar mar
   ```

   By representing the sentences as lists, the item indexes are not reliable: In the format description above, the token at position 1 is vamos and not vámonos. Thus to cope with this problem, we encode the sentences as **dictionaries**, where the keys are the id numbers, and this is resolved in the previously implemented function `convert_to_dict()`.

2. Some corpora have sentence numbers: This is solved by discarding lines starting with a `#`, as done in the CoNLL reader.

3. Extracting the pairs and triples: The implemented function `extract_pairs_and_triples()` extracts the nbest most frequent pairs and triples of a given corpus and returns two sorted lists of tuples: `frequent_pairs` and `frequent_triples`. As in the previous sections, we sort them by frequency and then by alphabetical order using `sorted()`.

Running the extractor on all the corpora

After reading the files and proccessing them to get the desired format, we extract nbest triples in the French GSD corpus, the Russian SynTagRus corpus, and the English EWT corpus. The results contain a list of tuples: `(subject, verb, object), freq)`. In the following, we report the three most frequent triples in the three languages respectively:

Using the French GSD corpus

```
[(('il', 'fait', 'partie'), 16),
 (('elle', 'fait', 'partie'), 7),
 (('il', 'comptait', 'habitants'), 7)]
```

Using the Russian SynTagRus corpus

```
[(('мы', 'имеем', 'дело'), 6),
 (('мы', 'имеем', 'что'), 4),
 (('мы', 'сделаем', 'все'), 4)]
```

Using the English EWT corpus

```
[((’you’, ’have’, ’questions’), 22),
 ((’you’, ’think’, ’what’), 12),
 ((’i’, ’do’, ’what’), 7)]
```

5 Resolving the entities

In this task, we need to extract the **relations** involving named entities, where both the subject and the object are **proper nouns**.

To solve this problem, we write an `extract_entity_triples(formatted_corpus_dict)` that process the corpus and return a list of `(subject, verb, object)` triples. Here we need to keep the original case and the triples are in the alphabetical order, for instance United States and not united states. Below The five first triples are obtained by running the implemented function:

```
[(’Baba’, ’remember’, ’George’),
 (’Beschta’, ’told’, ’Planet’),
 (’Boi’, ’beat’, ’Lopez’),
 (’Bush’, ’mentioned’, ’Arabia’),
 (’Bush’, ’mentioned’, ’Osama’)]
```

6 Reading

In this task, we need to read the article: *PRISMATIC: Inducing Knowledge from a Large Scale Lexicalized Relation Resource* by Fan and al. (2010) [pdf] and compare it to our work in this assignment.

In this paper [9], PRISMATIC is presented as a large scale **lexicalized relation resource** that is built automatically over 30 gb of text. This system is built using a suite of NLP tools that includes a dependency parser, a rule based named entity recognizer and a coreference resolution component. PRISMATIC is composed of **frames** which are the basic semantic representation of lexicalized relation and surrounding context.

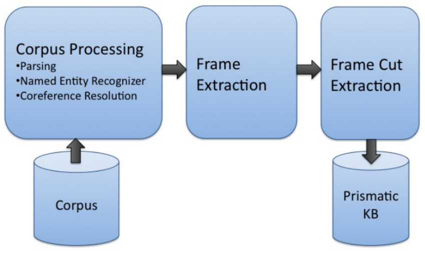


Figure 4: The PRISMATIC pipeline consists of three phases.

Compared to the manually created resources as the case in our assignment, PRISMATIC is an automatic process and has the following major characteristics:

- Corpus Processing: The key step in this stage is the application of a dependency parser which is used to identify the frame slots for the Frame Extraction stage.

- Frame Extraction: Frames are extracted based on the **dependency parses** and associated annotations.
- Frame-Cut Extraction: Frame-cuts of interest(e.g. S-V-O cuts) are identified over all frames and **frequency information** for each cut is tabulated. One of the main reasons for extracting a large amount of frame data from a corpus is to *induce interesting knowledge patterns* by exploiting **redundancy** in the data.
- The frames are defined by the words in a sentence and the relations between them.
- It uses a very small **set of slots** defined by parser and relation annotators to link a frame and its arguments.

7 Appendix

References

[1] Pierre Nugues, *Language processing with Perl and Prolog*, 2nd edition, 2014, Springer.

[2] P. Nugues, Language Technology - EDAN20, Lectures notes: `https://cs.lth.se/edan20/`

[3] D. Jurafsky, J. Martin, *Speech and Language Processing*, 3rd edition. Online: `https://web.stanford.edu/~jurafsky/slp3/`

[4] VanderPlas, A Whirlwind Tour of Python. O'Reilly 2016. Online reading: `https://jakevdp.github.io/WhirlwindTourOfPython/`

[5] Pierre Nugues, Assignment 5: `https://github.com/pnugues/edan20/blob/master/notebooks/5-triples.ipynb`

[6] Universal Dependencies: `https://universaldependencies.org/`

[7] Graphical representation of sentences, conllu.js: CoNLL-U format library for JavaScript: `http://spyysalo.github.io/conllu.js/`

[8] Language Pipelines. `http://vilde.cs.lth.se:9000/#/`

[9] J Fan, D Ferrucci and al. (2010), PRISMATIC: Inducing Knowledge from a Large Scale Lexicalized Relation Resource. IBM Watson Research Lab: `https://www.aclweb.org/anthology/W10-0915.pdf`

[10] Geron, Jupyter notebook on linear algebra from Machine Learning and Deep Learning in Python using Scikit-Learn, Keras and TensorFlow: `https://github.com/ageron/handson-ml2`