Language Technology - EDAN20
Assignment 4 - Fall 2020

Extraction of Subject-Verb-Object Triples

Hicham Mohamad, hi8826mo-s
hsmo@kth.se

August 30, 2020

1 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

2 Orgnization and locaton

The fourth lab session will take place on October 1st and 2nd. There can be last minute changes. Please always check the official times here:

`https://cloud.timeedit.net/lu/web/lth1/ri1Q5006.html`

You can work alone or collaborate with another student.

Each group will have to:

- Extract all the subject–verb pairs and subject–verb–object triples from a parsed corpus
- Rank the pairs and triples by frequency
- Optionally, create a simple **coreference solver** for the Swedish som 'who' pronoun.

3 Programming

This assignment is inspired by the **Prismatic** knowledge base used in the IBM Watson system. See this paper **Fan and Ferrucci et al**, 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.

3.1 Choosing a parsed corpus

1. 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**]. Local copies: [**training set**] [**test set**] [**test set with answers**]. You will use the training set only.

2. Draw a graphical representation of the two first sentences of the training set.
3. Download **What's wrong with my NLP** and use it to check your representations.
4. Apply the dependency parser for Swedish of the **Langforia pipelines** to these sentences. Link to Lanforia pipelines:

   `http://vilde.cs.lth.se:9000/`

3.2 Extracting the subject–verb pairs and subject–verb–object triples

You will extract all the subject–verb pairs and the subject–verb–object triples from the training corpus. To start the program, you can use the CoNLL-X reader available **here**. This program will enable you to read the other corpora. You can also program a reader yourself starting from the one you used to read the CoNLL 2000 format in the third lab or from scratch. You will find the description of the CoNLL-X format [**here**]. Archive **here**.

You can design the program as you want. However, here are some hints on the results:

- You will need to use Python's dictionaries. Be sure you understand them and note that you can use tuples as keys.
- Extract all the subject-verb pairs in the corpus. The subject function uses the SS code in the Swedish corpus of CoNLL-X. In the extraction, do not check the part of speech of the verb in the pair.
- Compute the total number of pairs. You should find 18885 pairs.
- Sort your pairs by order of frequency and give the five most frequent pairs. Here are the frequencies you should find:

  537
  261
  211
  171
  161

- Extract all the subject–verb–object triples of the corpus. The object function uses the OO code. Compute the total number of triples. You should find 5844 triples.
- Sort your triples by order of frequency and give the five most frequent pairs. Here are the frequencies you should find:

  37
  36
  36
  19
  17

Multilingual Corpora

Once your program is working on Swedish, you will apply it to all the other languages from this repository: **Universal Dependencies**. The address to download the corpora is **here**. You have a local version in the */usr/local/cs/EDAN20/* folder. You can read the training files from the folder using the CoNLL reader provided for the first part. You will need to adapt your program to CoNLL-U. See **here**.

Here are suggestions to carry out this task:

1. Some corpora expand some tokens into multiwords. This is the case in French, Spanish, and German. The table below shows examples of such expansions.

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.

In the corpora, you have the original tokens as well as the multiwords as with *vámonos al mar*.

```
1–2 ávmonos _
1 vamos ir
2 nos nosotros
3–4 al _
3 a a
4 el el
5 mar mar
```

Read the format description for the details: [**CoNLL-U format**].

2. If you represent the sentences as lists, the item indexes are not reliable: In the format description, the token at position 1 is vamos and not vámonos. You have two ways to cope with this:

   - Either remove all the lines that include a range in the id field.
   - or encode the sentences as dictionaries (preferable), where the keys are the id numbers. Here are the results for the second sentence of the Swedish CoNLL X corpus:

```
{'5': {'pdeprel': '_', 'lemma': '_', 'postag': 'NN', 'phead': '_', '
    feats': '_', 'form':
'dagens', 'id': '5', 'deprel': 'DT', 'cpostag': 'NN', 'head': '6'},
'7': {'pdeprel': '_', 'lemma': '_', 'postag': 'IP', 'phead': '_', '
    feats': '_', 'form': '.',
'id': '7', 'deprel': 'IP', 'cpostag': 'IP', 'head': '1'},
'3': {'pdeprel': '_', 'lemma': '_', 'postag': 'TP', 'phead': '_', '
    feats': '_', 'form':
'äberttigad', 'id': '3', 'deprel': 'SP', 'cpostag': 'TP', 'head': '1
    '},
'1': {'pdeprel': '_', 'lemma': '_', 'postag': 'AV', 'phead': '_', '
    feats': '_', 'form':
'Är', 'id': '1', 'deprel': 'ROOT', 'cpostag': 'AV', 'head': '0'},
'6': {'pdeprel': '_', 'lemma': '_', 'postag': 'NN', 'phead': '_', '
    feats': '_', 'form':
'äsamhlle', 'id': '6', 'deprel': 'PA', 'cpostag': 'NN', 'head': '4'
    },
'2': {'pdeprel': '_', 'lemma': '_', 'postag': 'PO', 'phead': '_', '
    feats': '_', 'form':
'den', 'id': '2', 'deprel': 'SS', 'cpostag': 'PO', 'head': '1'},
'0': {'pdeprel': 'ROOT', 'lemma': 'ROOT', 'postag': 'ROOT', 'phead':
     '0', 'feats': 'ROOT',
'form': 'ROOT', 'id': '0', 'deprel': 'ROOT', 'cpostag': 'ROOT', '
    head': '0'},
 '4': {'pdeprel': '_', 'lemma': '_', 'postag': 'PR', 'phead': '_', '
     feats': '_', 'form': 'i',
'id': '4', 'deprel': 'RA', 'cpostag': 'PR', 'head': '1'}}
```

3. Some corpora have sentence numbers. Here you solve it by discarding lines starting with a `#`. This is already done in the CoNLL reader.

4. All the corpora in the universal dependencies format use the same function names: `nsubj` and `obj` for the subject and direct object.

5. Run your extractor on all the corpora and report the five most frequent pairs and triples you obtained in three languages. Preferably, languages you can decipher so that you can check the tuples are relevant. Otherwise, use Google Translate.

4 Reading

Read the article: *PRISMATIC: Inducing Knowledge from a Large Scale Lexicalized Relation Resource* by Fan and al. (2010) [pdf] and write in a few sentences how it relates to your work in this assignment.

5 Solving coreferences (Optional)

In this part, you will resolve a simple **anaphor** involving the Swedish *som* 'who' pronoun.

- Use What's wrong with my NLP to derive a simple rule to find the antecedent of *som*
- Replace all the occurrences of *som* as a subject with its antecedent in your pairs and triples.

6 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 4: `https://github.com/pnugues/edan20/blob/master/notebooks/4-chunker.ipynb`

[6] Chunking: `https://www.clips.uantwerpen.be/conll2000/chunking/`

[7] Taku Kudoh, Yuji Matsumoto, Use of Support Vector Learning for Chunk Identification: `https://www.aclweb.org/anthology/W00-0730.pdf`

[8] YamCha: Yet Another Multipurpose Chunk Annotator. `http://www.chasen.org/~taku/software/yamcha/`

[9] Akbik et al., Contextual String Embeddings for Sequence Labeling, 2018.

[10] Zalando Research, Github code and pretrained language models: `https://github.com/zalandoresearch/flair`

[11] Tutorial: Web Scraping with Python Using Beautiful Soup: `https://www.dataquest.io/blog/web-scraping-tutorial-python/`

[12] 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`