

A practical workshop on automatic morpho-syntactic annotation of large language corpora using the Universal Dependencies framework

17th of April 2024 University of Tartu, Estonia

Hanna-Mari Kupari hmknie@utu.fi

➤ Workshop Day 3

➤ Recap of: A short introduction to machine learning

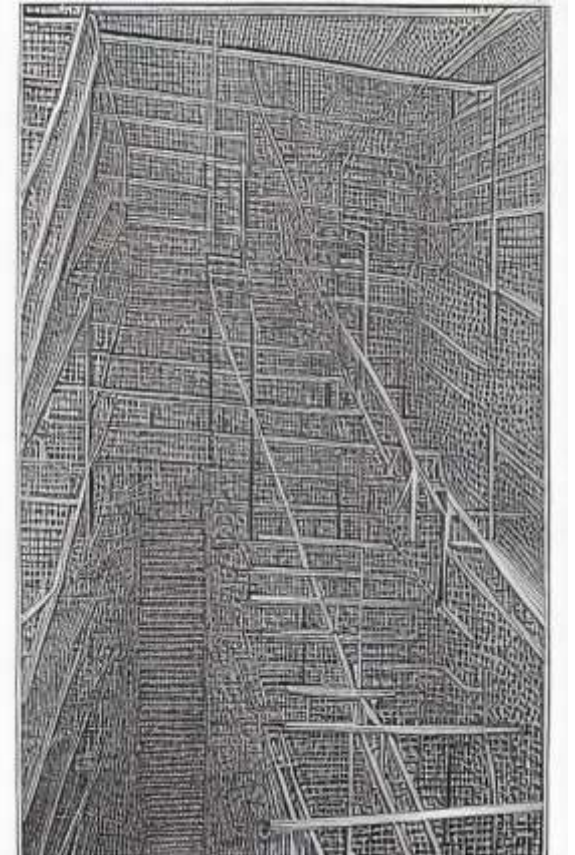
What is a bag of words?

What are word embedding?

What are possible tools for parsing your own data?

Working with files in Stanza and Trakit

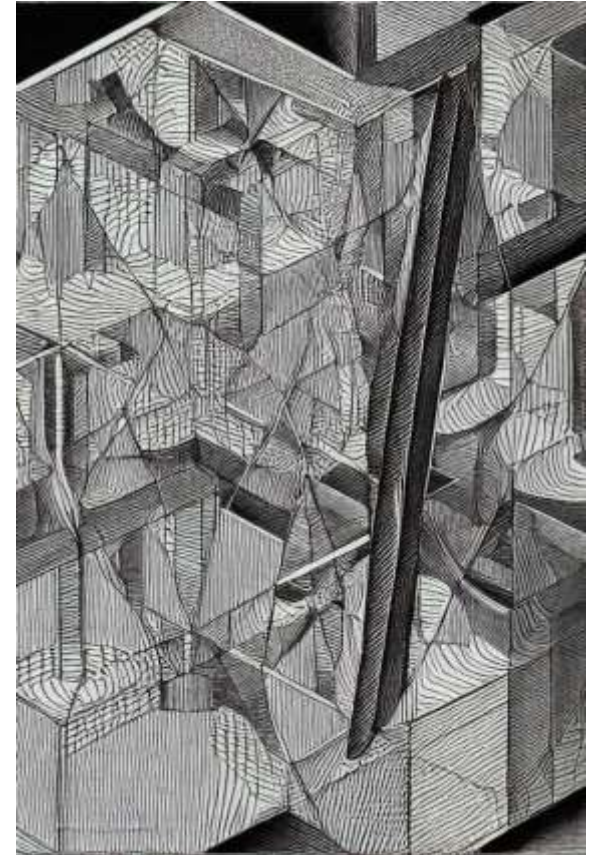
Personal projects help



Deep AI

Shortly about today's aims

- Working in a workshop style
- Some basic points from the lecture on Tuesday
- How to understand word embeddings?
- Continued: Two tools to work with: Trankit and Stanza





Recap (and introductions)

- Say your name (so I'll try and remember)
- What do you remember from yesterday's lecture?
- Any inspirations or ideas from Tuesday (Monday)?

Recap

- We should now be familiar with **concepts** like
 - Dependency grammar
 - Treebanks
 - CoNLL-U format
- And used to working with **tools** like
 - UD pipe 2 online demo
 - Stanza in a Jupyter-notebook

Case study: penitentiary documents

- A case study combining parsers with traditional close reading
- A search in POS column with POS label VERB and taking unique LEMMAS
- Close reading this dataset and picking verbs related to violence (e.g. to hit)
- Taking these sentences and adding semantic roles (BRAT-tool)
- The annotation can be kept consistent and tidy with digital methods

BRAT

annotation tool

https://brat.nlpab.org/introduction.html


home | introduction | examples | features | manual | site map | contact | brat

mini-introduction to brat

brat is a web-based tool for text annotation; that is, for adding notes to existing text documents.

brat is designed in particular for structured annotation, where the notes are not freeform text but have a fixed form that can be automatically processed and interpreted by a computer.

the following screenshot shows a simple example where a sentence has been annotated to identify mentions of some real-world entities (things) and their types, and a relation between two.




example annotations (following in part the [ACE 2005](#) entity and relation annotation guidelines)

this example illustrates two basic categories of annotation:

- **text span** annotations, such as those marked with the *Organization* and *Person* types in the example
- **relation** annotations, such as the *family* relation in the example

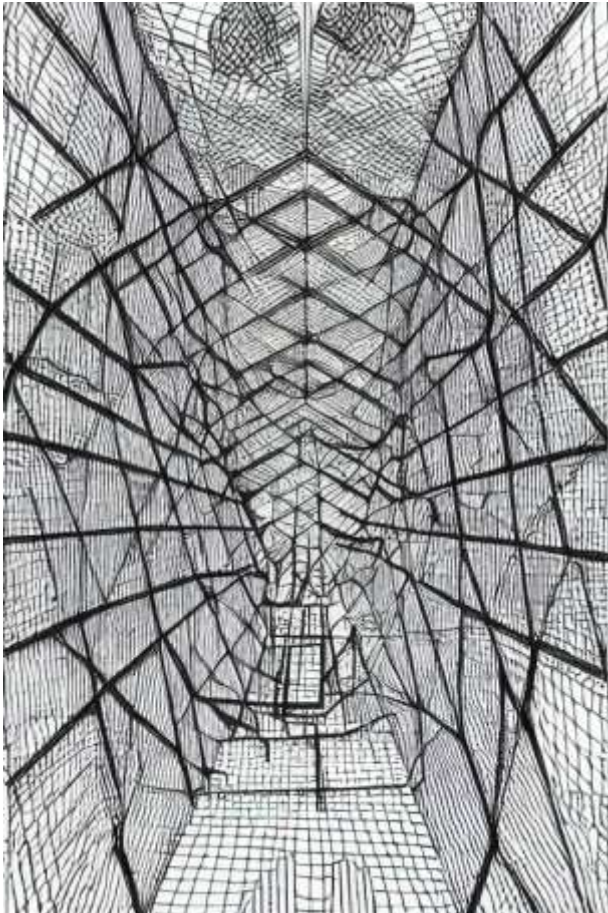
the simple typed text span category is suitable for creating annotations for [named entity recognition](#), and binary relations for simple relational [information extraction](#) tasks, among others.

brat also supports the annotation of **n-ary associations** that can link together any number of other annotations participating in specific roles. This category of annotation can be used for example for event annotation, such as *TRANSFER* in the following example:



Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou and Jun'ichi Tsujii (2012). brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations Session at EACL 2012*.

Modern parsing tools



- Based on machine learning
- Natural language processing uses the two types
 - Supervised machine learning
 - Unsupervised

Some practical examples

- Supervised
- Based on given labels
- A **parser** trained on Gold Standard training data
- Named-Entity Recognition

- Unsupervised
- Based on statistics
- **Topic modeling**
 - **is a type of statistical modeling used to identify topics or themes within a collection of documents.** It involves automatically clustering words that tend to co-occur frequently across multiple documents, with the aim of identifying groups of words that represent distinct topics.
(upenn.edu)

Representing language as numbers

- What is a **bag of words** approach?
- Creating a sparse matrix out of vocabulary in corpus
- A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:
 1. A vocabulary of known words.
 2. A measure of the presence of known words.
- <https://machinelearningmastery.com/gentle-introduction-bag-words-model/>
- *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Jufarsky 2014

What are embeddings?

- words and documents are represented in the form of numeric vectors allowing similar words to have similar vector representations
- <https://www.turing.com/kb/guide-on-word-embeddings-in-nlp>
- XLM-Roberta Large is the basis of Trankit
- *XLM-RoBERTa* model pre-trained on 2.5TB of filtered CommonCrawl data containing 100 languages
- <https://huggingface.co/FacebookAI/xlm-roberta-large>

Today's main focus: HOW 2

- Advanced:
 - Trankit
 - Evaluation with a ready-made tool
- Beginners:
 - Stanza working with files
 - UD pipe 2 on web interface
- Main idea is to choose a tool (or do a comparison between the models?)

Advanced:

- If you attended Tuesday's lecture
- Work with the files you have and go on towards automatic evaluation

<https://universaldependencies.org/conll18/evaluation.html>

CoNLL 2018 Shared Task

Home
Proceedings | Program (Schedule)
Data | Baseline Models
Evaluation | TIRA | Results
EPE 2018 | EPE Results
Paper Submission Guidelines
Organization | Registration | Timeline

CONTACT

udat-org@googlegroups.com

NEWS

October 28: System outputs published
September 13: Proceedings
September 4: Results of old systems
July 2: Test data made public
July 2: **MAIN RESULTS**
June 25: Eval script 1.2
June 21: Test phase extended to July 1
June 17: Baseline results
June 1: Test phase started
May 2: Baseline models
April 15: **TRAINING DATA**
March 6: Trial data

Evaluation

The evaluation script (2018 version) is available for [download here](#).

All systems will be required to generate valid output in the CoNLL-U format for all test sets.

The definition of "valid" is not as strict as for released UD treebanks. For example, an unknown dependency relation label will only cost one point in the labeled accuracy score but it will not render the entire file invalid. However, cycles, multiple root nodes, wrong number of columns or wrong indexing of nodes will be considered invalid output. The score on the particular test file will then be set to 0 (but the overall macro-score will still be non-zero if outputs for other test files are valid).

The systems will know the language and treebank code of the test set, but they must respond even to unknown language/treebank codes (for which there are no training data). The systems will be able to select either raw text as input, or the file pre-processed by UDPipe. Every system must produce valid output for every test set.

Several different metrics, evaluating different aspects of annotation, will be computed and published for each system output. Three main system rankings will be based on three main metrics. None of them is more important than the others and we will not combine them into a single ranking. Participants who want to decrease task complexity may concentrate on improvements in just one metric; however, all participating systems will be evaluated with all three metrics, and participants are strongly encouraged to output all relevant annotation (syntax + morphology + lemmas), even if they just copy values predicted by the baseline model. The three main metrics are:

- **LAS** (labeled attachment score) will be computed the same way as in the 2017 task so that results of the two tasks can be compared.
- **MLAS** (morphology-aware labeled attachment score) is inspired by the CLAS metric

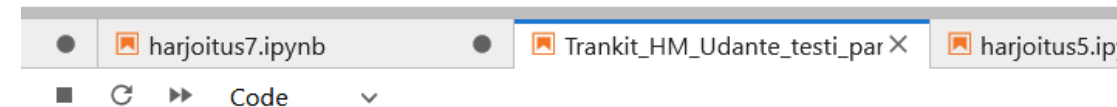
Evaluation using Conll18

- <https://universaldependencies.org/conll18/evaluation.html>
- Read the docs
- See how and much the F1 scores differ from parser to parser
- Look for existing publications on how these might be improved

Advanced level

- Use Trankit
- Or perhaps even train Trankit?
 - Combining the existing UD treebanks to form new training data
- Needs some computational power, I have used CSC Puhti

/node/r18c01.bullx/61715/lab/tree/kupariha/trankit/HM_udante/Trankit_HM_Udante_testi_par



```
<it
```

```
object_2008402/kupariha/venv/lib64/python3.9/site-packages/tqdm/auto.py:21: TqdmW  
tps://ipywidgets.readthedocs.io/en/stable/user_install.html  
notebook import tqdm as notebook_tqdm
```

```
ify_customized_pipeline(  
    lang='customized-mwt', # pipeline category  
    cache_dir='./save_dir', # directory used for saving models in previous steps  
    lang_name='xlm-roberta-base' # embedding version that we use for training our cust
```

```
    \line is ready to use!  
    \alized as follows:
```

```
-----  
t import Pipeline  
e(lang='customized-mwt', cache_dir='./save_dir')
```

```
https://raw.githubusercontent.com/HannaKoo/Latin-variability/main/harmonization/f
```

```
udante-ud-test.conllu > test.conllu
```

```
/conllu_to_text.pl --lang la < test.conllu > test.txt
```

```
_udante-ud-test.conllu' already there; not retrieving.
```

```
actions for processing the trankit output format
```

```
ine(token):  
t".join([str(token["id"]), token["text"], token.get("lemma", "_"), token.get("up  
ne
```

```
llu(d):
```

```
in ("test_output_conllu", "w", encoding="UTF8") as f:
```

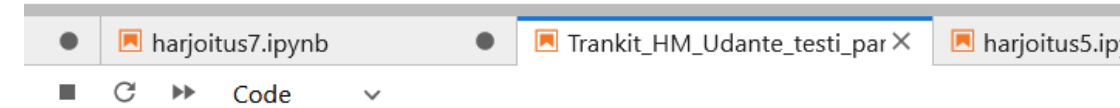
Using Trankit

- https://github.com/HannaKoo/ParsersTartu/blob/main/Trankit_use.md
- <http://nlp.uoregon.edu/trankit> (For beginners)

Beginner's level

- Use Stanza to work with files
- Or continue with just several sentences to feed in manually
- Starting point a Github MD document
- Working together step by step

/node/r18c01.bullx/61715/lab/tree/kupariha/trankit/HM_udante/Trankit_HM_Udante_testi_par



```
<it
```

```
object_2008402/kupariha/venv/lib64/python3.9/site-packages/tqdm/auto.py:21: TqdmW
tps://ipywidgets.readthedocs.io/en/stable/user_install.html
nernotebook import tqdm as notebook_tqdm
```

```
ify_customized_pipeline(  
    category='customized-mwt', # pipeline category  
    save_dir='./save_dir', # directory used for saving models in previous steps  
    lang_name='xlm-roberta-base' # embedding version that we use for training our cust
```

```
    \line is ready to use!  
    \alized as follows:  
    .....
```

```
from Pipeline  
p = Pipeline(lang='customized-mwt', cache_dir='./save_dir')
```

```
https://raw.githubusercontent.com/HannaKoo/Latin-variability/main/harmonization/f
```

```
udante-ud-test.conllu > test.conllu
```

```
/conllu_to_text.pl --lang la < test.conllu > test.txt
```

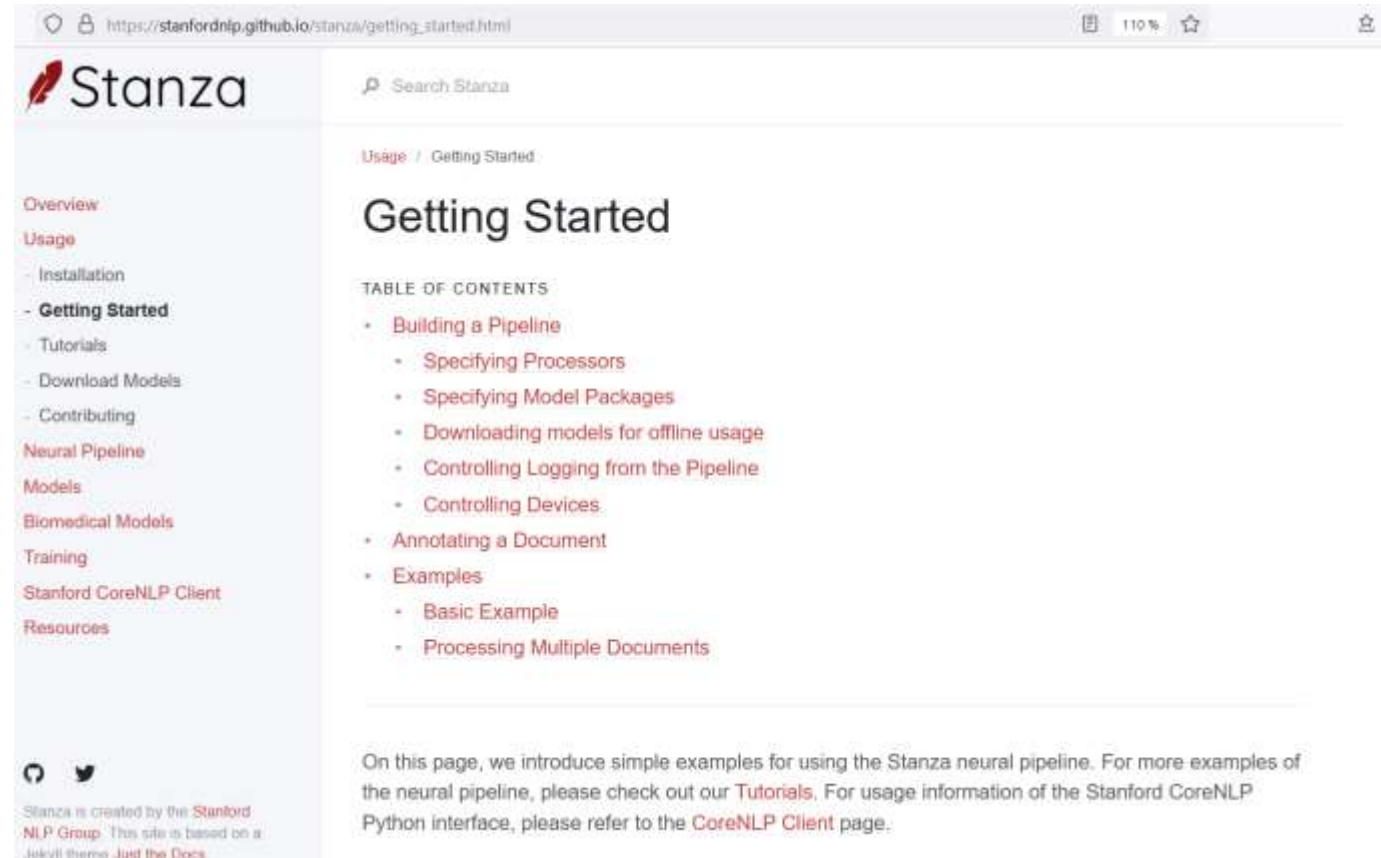
```
'udante-ud-test.conllu' already there; not retrieving.
```

```
actions for processing the trankit output format
```

```
def line(token):  
    t = token.join([str(token["id"]), token["text"], token.get("lemma", "_"), token.get("up  
    re
```

```
def conllu(d):  
    p = ("test_output_conllu" + "u" + encoding="UTF8") as f:
```

Stanza



The screenshot shows a web browser displaying the Stanza Getting Started page. The browser's address bar shows the URL `https://stanfordnlp.github.io/stanza/getting_started.html`. The page has a light blue sidebar on the left with a search bar and a navigation menu. The main content area is white and features a breadcrumb trail, a title, a table of contents, and a paragraph of introductory text.

Stanza

Search Stanza

Usage / Getting Started

Getting Started

TABLE OF CONTENTS

- [Building a Pipeline](#)
 - [Specifying Processors](#)
 - [Specifying Model Packages](#)
 - [Downloading models for offline usage](#)
 - [Controlling Logging from the Pipeline](#)
 - [Controlling Devices](#)
- [Annotating a Document](#)
- [Examples](#)
 - [Basic Example](#)
 - [Processing Multiple Documents](#)

On this page, we introduce simple examples for using the Stanza neural pipeline. For more examples of the neural pipeline, please check out our [Tutorials](#). For usage information of the Stanford CoreNLP Python interface, please refer to the [CoreNLP Client](#) page.

Stanza is created by the [Stanford NLP Group](#). This site is based on a [Jekyll](#) theme [Just the Docs](#).

Using Stanza step by step beginner's level

- Have a look at <https://stanfordnlp.github.io/stanza/>
- Open the .md file from https://github.com/HannaKoo/ParsersTartu/blob/main/Stanza_use.md
- We will first work with English and then:
- <https://stanfordnlp.github.io/stanza/performance.html#system-performance-on-ud-treebanks>
- Working with files:
- https://github.com/HannaKoo/ParsersTartu/blob/main/documents_use



Kieli- ja käännöstieteiden laitos 2024