Morpho-syntactic Analysis with the *Stanford CoreNLP* *spaCy* and *Stanza*

Danilo Croce
croce@info.uniroma2.it

Web Mining & Retrieval

2021

Objectives of this tutorial

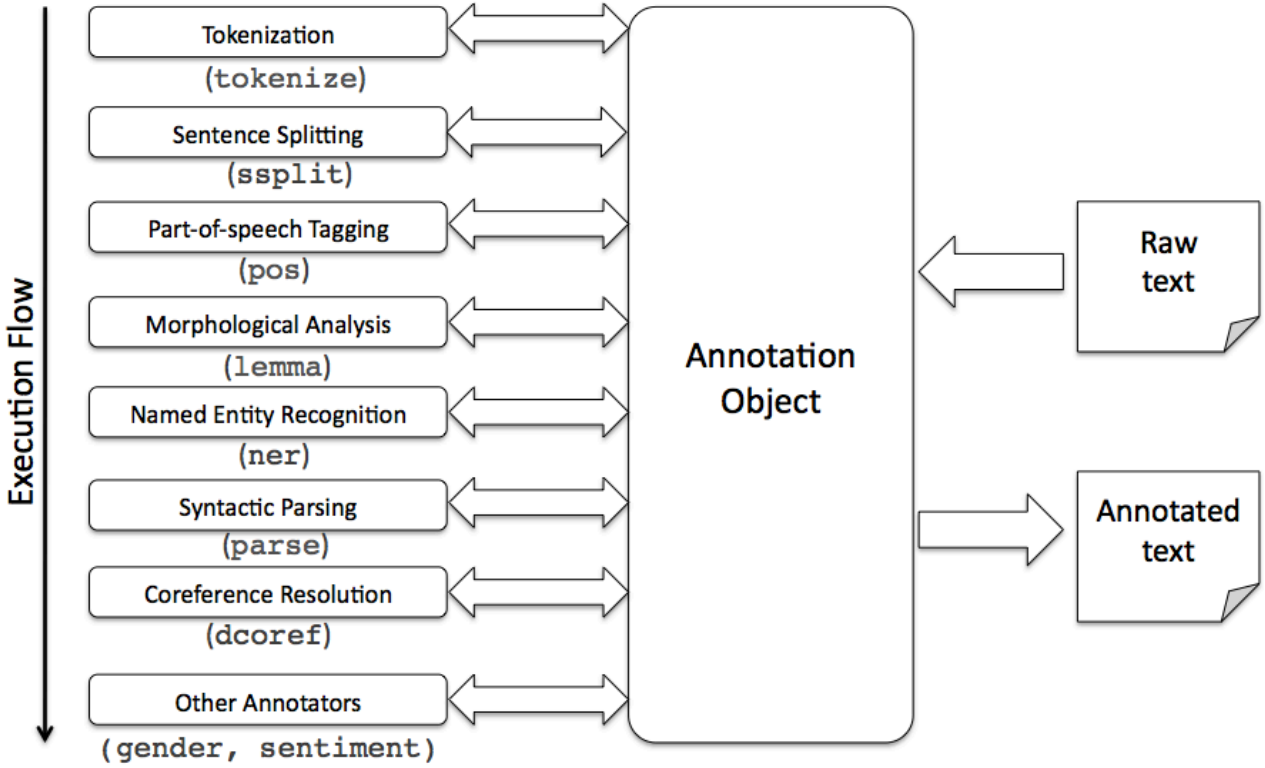
- Use of a Natural Language Toolkit
  - CoreNLP toolkit
- Morpho-syntactic analysis of short texts
  - Tokenization
  - Part-of-speech Tagging
  - Lemmatization
  - Named Entity Recognition
  - Dependency Parsing
- Use of such body of linguistic evidence for a task
  - Knowledge Acquisition by reading large scale corpora

The Stanford CoreNLP

- The Stanford CoreNLP is a statistical natural language parser from the Stanford Natural Language Processing Group.
  - Used to parse input data written in several languages
    - such as English, German, Arabic and Chinese
- it has been developed and maintained since 2002, from the Stanford University
  - Written in Java
  - The application is licensed under the GNU GPL, but commercial licensing is also available.
- Site: `http://stanfordnlp.github.io/CoreNLP/`
- A demo is available at: `http://corenlp.run/`

Stanford CoreNLP

- The list of processors



An example

- Let us consider the sentence:
  - *"In 1982, Mark drove his car from Los Angeles to Las Vegas."*

Part-of-Speech:

	IN	CD	,	NNP	VBD	PRP$	NN	IN	NNP	NNP	TO	NNP	NNP	.
1	In	1982	,	Mark	drove	his	car	from	Los	Angeles	to	Las	Vegas	.

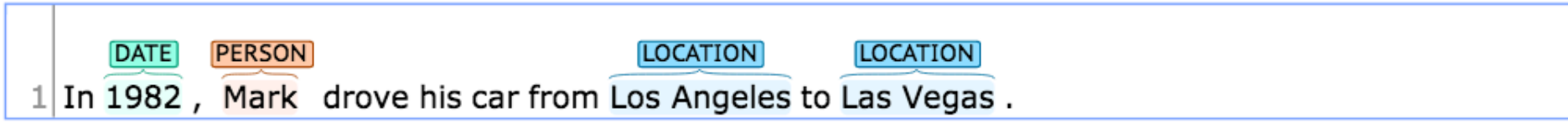
Lemmas:

	in	1982	,	Mark	drive	he	car	from	Los	Angeles	to	Las	Vegas	.
1	In	1982	,	Mark	drove	his	car	from	Los	Angeles	to	Las	Vegas	.

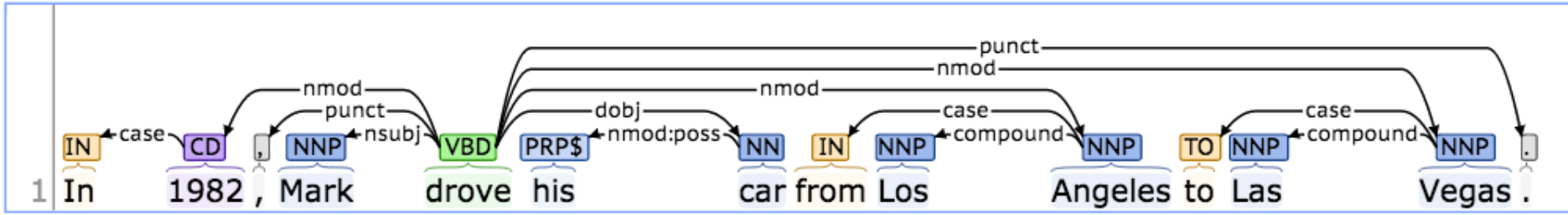
An example

- Let us consider the sentence:
  - *"In 1982, Mark drove his car from Los Angeles to Las Vegas."*

Named Entity Recognition:



Basic Dependencies:



How to use from Java

- It can be easily integrated in a Java project that support Maven, adding the following dependency to the pom.xml file

```
<dependencies>
        <dependency>
                <groupId>edu.stanford.nlp</groupId>
                <artifactId>stanford-corenlp</artifactId>
                <version>3.6.0</version>
        </dependency>
        <dependency>
                <groupId>edu.stanford.nlp</groupId>
                <artifactId>stanford-corenlp</artifactId>
                <version>3.6.0</version>
                <classifier>models</classifier>
        </dependency>
</dependencies>
```

A small example in JAVA

```
// We see here how to parse a sentence and write on

// The Properties object contains the list of processors to be loaded
Properties props = new Properties();
props.put("annotators", "tokenize, ssplit, pos, lemma, ner, parse");

// Intializing the StanfordCoreNLP toolkit
```
StanfordCoreNLP pipeline = **new StanfordCoreNLP(props);**
String str = "In 1982, Mark drove his car from Los Angeles to Las Vegas";

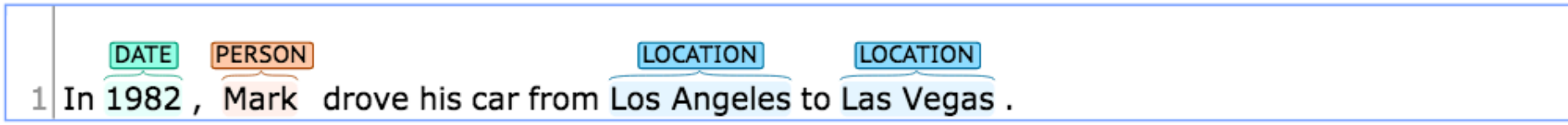
The CONLL format

```
Annotation document = new Annotation(str);
pipeline.annotate(document);
List<CoreMap> sentences = document.get(SentencesAnnotation.class);
// for each sentence
for (CoreMap sentence : sentences) {
        // Extract the dependency graph
        SemanticGraph dependencies = sentence.get(BasicDependenciesAnnotation.class);
        // For each token
        for (CoreLabel token : sentence.get(TokensAnnotation.class)) {
                IndexedWord sourceNode = dependencies.getNodeByIndex(token.index());
                IndexedWord father = dependencies.getParent(sourceNode);
                // If no father is available, the token is associated to the "root" node
                int fatherId = 0;
                String relName = "root";
                if (father != null) {
                        fatherId = father.index();
                        SemanticGraphEdge edge = dependencies.getEdge(father, sourceNode);
                        relName = edge.getRelation().getShortName();
                }
                System.out.println(token.index() + "\t" + token.originalText() +
                        "\t" + token.lemma() + "\t" + token.tag()+ "\t" +
                        token.ner() + "\t" + relName + "\t" + fatherId);
        }
        System.out.println();
}
```

The CONLL tabular format

Token ID	Surface	Lemma	POS	Namend Entity	Relation Name	Dependant
1	In	in	IN	O	case	2
2	1982	1982	CD	DATE	rmod	5
3	,	,	,	O	punct	5
4	Mark	Mark	NNP	PERSON	nsubj	5
5	drove	drive	VBD	O	root	0
6	his	he	PRP$	O	rmod:poss	7
7	car	car	NN	O	dobj	5
8	from	from	IN	O	case	10
9	Los	Los	NNP	LOCATION	compound	10
10	Angeles	Angeles	NNP	LOCATION	rmod	5
11	to	to	TO	O	case	13
12	Las	Las	NNP	LOCATION	compound	13
13	Vegas	Vegas	NNP	LOCATION	rmod	5
14	.	.	.	O	punct	5

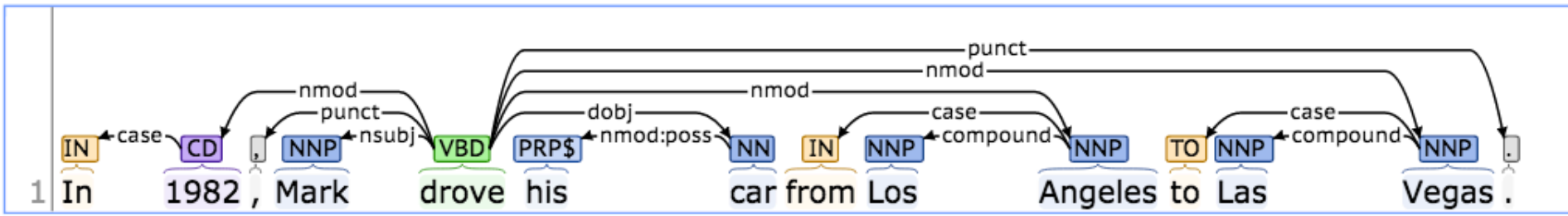
Named Entity Recognition:



The CONLL tabular format

Token ID	Surface	Lemma	POS	Namend Entity	Relation Name	Dependant
1	In	in	IN	O	case	2
2	1982	1982	CD	DATE	nmod	5
3	,	,	,	O	punct	5
4	Mark	Mark	NNP	PERSON	nsubj	5
5	drove	drive	VBD	O	root	0
6	his	he	PRP$	O	nmod:poss	7
7	car	car	NN	O	dobj	5
8	from	from	IN	O	case	10
9	Los	Los	NNP	LOCATION	compound	10
10	Angeles	Angeles	NNP	LOCATION	nmod	5
11	to	to	TO	O	case	13
12	Las	Las	NNP	LOCATION	compound	13
13	Vegas	Vegas	NNP	LOCATION	nmod	5
14	.	.	.	O	punct	5

Basic Dependencies:



Spacy - "Industrial-Strength" NLP Library

- A Natural Language processing Pipeline in Python
  - Fastest in the world written in Cython
  - Get things done easy to install simple API

- https://spacy.io/

How to use spaCy

- Installation:
- Install anaconda: https://docs.anaconda.com/anaconda/install/

```
$ conda create -n spacyenv
$ conda activate spacyenv
$ conda install spacy
$ python -m spacy download en
```

- Load model and process text:

```
import spacy
nlp = spacy.load('en')
doc = nlp('Can you process this text?')
```

How to parse a sentence

```
import spacy

nlp = spacy.load('en')
doc = nlp('In 1982, Mark drove his car from Los Angeles to Las Vegas')


for sent in doc.sents:
     for i, word in enumerate(sent):
         if word.head is word:
             head_idx = 0
         else:
            head_idx = doc[i].head.i+1

         entity_tag = word.ent_type_
         if len(entity_tag) == 0:
             entity_tag = "O"

         print("%d\t%s\t%s\t%s\t%s\t%s\t%d"%(i+1, word, word.lemma_, word.tag_,
             entity_tag, word.dep_, head_idx))
```

Final objective of this exercitation

- Let us try to navigate the parse tree
- It will provide a first form of Ontology Learning by exploiting Syntactic information
  - Acquiring simple facts from the automatic analysis of large scale corpora, e.g.

```
PERSON – marry – PERSON
```

- We will extract morpho-syntactic parser in the form

```
Subject – VERB – Direct Object
```

- We need a large-scale corpus already processed with CoreNLP

Stanza

- Another Natural Language processing Pipeline in Python
  - Released by the Stanford Group
- https://stanfordnlp.github.io/stanza/index.html

- A small tutorial for using Stanza and Spacy
- https://colab.research.google.com/drive/1GGMBlKJFyytBqvlgUxuHEyqEU2rcjb3y?usp=sharing

How to parse a sentence with Stanza

- To install Stanza

```
! pip install stanza
```

- To parse a sentence

```
import stanza
stanza.download('en') # download English model

nlp = stanza.Pipeline('en', processors='tokenize,mwt,pos,lemma,depparse,ner')

input_string = "In 1982, Mark drove his car from Los Angeles to Las Vegas"

doc = nlp(input_string)

for sent in doc.sentences:
     for word in sent.words:
         print("%d\t%s\t%s\t%s\t%s\t%s\t%d"%(word.id, word.text, word.lemma,
word.pos, sent.tokens[word.id-1].ner, word.deprel, word.head))
```

You will be provided with…

- A "small" selection of abstracts from Wikipedia processed with CoreNLP, already in CONLL format
  - 375K sentences
  - 9.6M of tokens
- A python script (`CoreNLP_simple_reader.py`) to
  - load the processed sentences stored as textual files in the CONLL format
  - navigate the resulting dependency graph
  - extract simple information

Exercise

1. Extract and count the occurrences of Nouns in the provided corpus
   - The list of Part-of-speech tags is provided here: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

2. Extract and count the occurrences of mentions to Named Entities in the provided corpus
   - Focus on the `PERSON` category

Exercise (2)

3. Extract and count all patterns in the form

```
Noun – nsubj – Verb – dobj – Noun
```

keeping the lemma of the involved nouns / verbs, e.g.,

```
Mark –      drive  –      car
```

4. Extract and count all patterns in the form extracted at step 3, but replace the Named Entities with their types

```
PERSON      –      drive  –      car
```

The complete list of existing dependency can be found here:

`http://nlp.stanford.edu/pubs/USD_LREC14_paper_camera_ready.pdf`

Useful Links

- CoreNLP Web Page:
  - `http://stanfordnlp.github.io/CoreNLP/`
- How to Download the last version of CoreNLP:
  - `http://nlp.stanford.edu/software/stanford-corenlp-full-2015-12-09.zip`
- Demo
  - `http://corenlp.run/`
- A description of CoreNLP:
  - `http://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf`

Alphabetical list of POS tags used in the Penn Treebank Project

Tag	Description	Tag	Description
CC	Coordinating conjunction	PRP$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential there	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	to
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VBN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non-3rd person singular present
NNP	Proper noun, singular	VBZ	Verb, 3rd person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb