Natural Language Processing

SoftLang Team, University of Koblenz-Landau

Prof. Dr. Ralf Lämmel
Msc. Johannes Härtel
Msc. Marcel Heinz

Natural Language Processing (NLP)

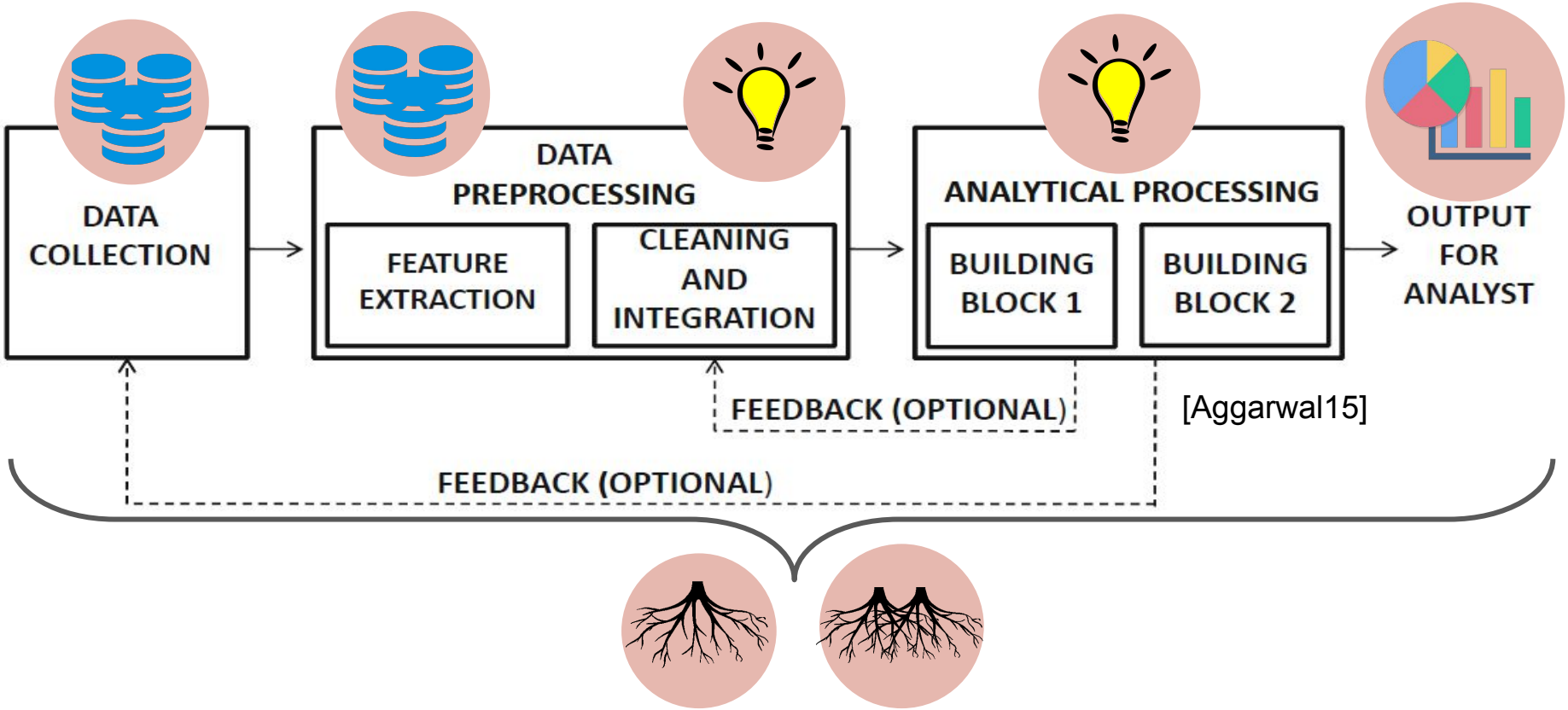
https://nlp.stanford.edu/software/lex-parser.shtml

`A natural language parser is a program that works out the grammatical **structure of sentences**, for instance, which groups of words go together (as "phrases") and which words are the **subject** or **object** of a verb.’

NLP Tasks

- Text translation
- Morphological analysis
- **Part-of-speech tagging**
- **Relationship extraction**
- **Text Similarity**
- **Topic Modeling**
- Natural language search
- Proofreading
- Question answering
- Natural language generation
- Text-to-speech
- Text simplification
- Word sense disambiguation

Back to the ‘Big Picture’



[Aggarwal15]

Noun Frequency Exploration

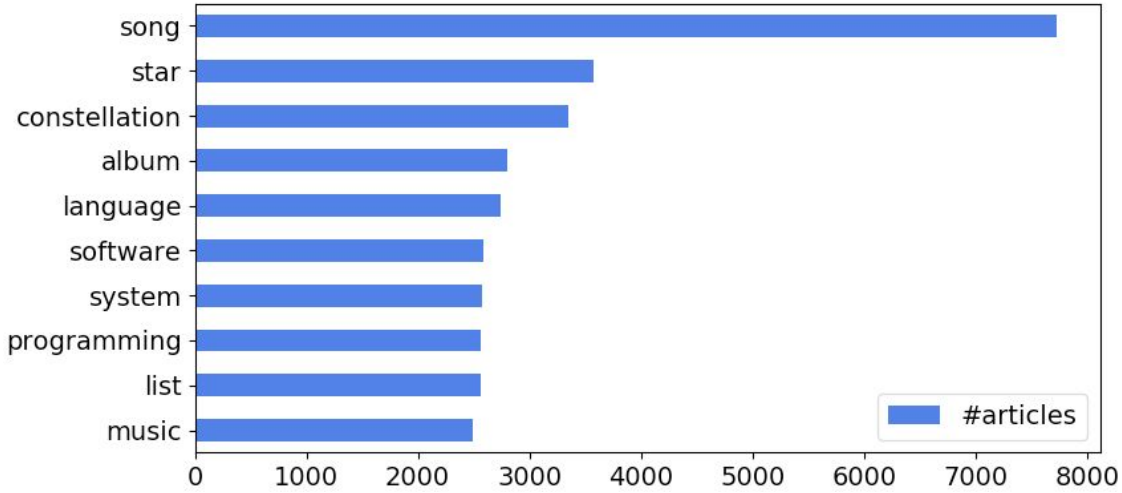
- In Wikipedia, we analyzed articles' summaries below the categories `Formal languages', `Computer file format' and `Installation Software'.

What are the most frequent nouns?

Noun Frequency Exploration

- In Wikipedia, we analyzed articles' summaries below the categories `Formal languages', `Computer file format' and `Installation Software'.

What are the most frequent nouns?



Noun Frequency Exploration

- In Wikipedia, we analyzed articles' summaries below the categories `Formal languages', `Computer file format' and `Installation Software'.

What are the most frequent nouns?

Lessons learnt:

- Typical research approach consists of the following steps: **Question** > **Hypothesis** > **Study** > **Analysis** > **Evaluation**
- Matching the approach to data mining:
  - State a **question** on what you intent to gain from an analysis? (What features do you need? What kind of analysis do you want to perform?)
  - State a **hypothesis** on the results of analysis. What will your results be? (What content do you expect?)
  - In the **study** part, you need to get an overview on your data. Does the content differ from your expectations? Are there missing preprocessing steps?
  - In the **analysis**, you have insights on the expected content. What do you use extracted features for?
  - In the **evaluation**, you need to provide qualitative insights on your analysis. Is your analysis correct? Use sampling methods.

Natural Language Processing (NLP)

A lot of resources are necessary for processing natural language, such as text corpora for training, tokenization models or stop word lists.

In [2]:
```
import nltk
from nltk.corpus import reuters

nltk.download('reuters')
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package reuters to
[nltk_data]     C:\Users\Johannes\AppData\Roaming\nltk_data...
[nltk_data]   Package reuters is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Johannes\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Johannes\AppData\Roaming\nltk_data...
```

Natural Language Processing (NLP)

The Reuters corpus contains thousands of texts from news feeds that can be used for training. Such corpora are accessed through the corpus API.

```
In [3]: reuters.raw('training/9865')

Out[3]: "FRENCH FREE MARKET CEREAL EXPORT BIDS DETAILED\
n  French operators have requested licences\n  t
o export 675,500 tonnes of maize, 245,000 tonnes
 of barley,\n  22,000 tonnes of soft bread wheat
 and 20,000 tonnes of feed\n  wheat at today's E
uropean Community tender, traders said.\n      R
ebates requested ranged from 127.75 to 132.50 Eu
ropean\n  Currency Units a tonne for maize, 136.
00 to 141.00 Ecus a tonne\n  for barley and 134.
25 to 141.81 Ecus for bread wheat, while\n  reba
tes requested for feed wheat were 137.65 Ecus, t
hey said.\n  \n\n"
```

Natural Language Processing (NLP)

Preprocessing for working with natural language artifacts are sentence tokenization ...

In [13]:
```
from nltk import sent_tokenize

text = reuters.raw('training/9865')
sentences = sent_tokenize(text, language='english')
for sentence in sentences:
    print("# "+sentence)
```

```
# FRENCH FREE MARKET CEREAL EXPORT BIDS DETAILED
 French operators have requested licences
 to export 675,500 tonnes of maize, 245,000 tonnes of barley,
 22,000 tonnes of soft bread wheat and 20,000 tonnes of feed
 wheat at today's European Community tender, traders said.
# Rebates requested ranged from 127.75 to 132.50 European
 Currency Units a tonne for maize, 136.00 to 141.00 Ecus a tonne
 for barley and 134.25 to 141.81 Ecus for bread wheat, while
 rebates requested for feed wheat were 137.65 Ecus, they said.
```

Natural Language Processing (NLP)

… word tokenization …

In [17]:
```
from nltk import word_tokenize

for sentence in sentences:
    print(word_tokenize(sentence))
```

```
['FRENCH', 'FREE', 'MARKET', 'CEREAL', 'EXPORT', 'BIDS', 'DETAILED', 'French', 'operators', 'have', 'requested', 'licences', 'to', 'export',
'675,500', 'tonnes', 'of', 'maize', ',', '245,000', 'tonnes', 'of', 'barley', ',', '22,000', 'tonnes', 'of', 'soft', 'bread', 'wheat', 'and', '20,000', 'tonnes',
'of', 'feed', 'wheat', 'at', 'today', "'s", 'European', 'Community', 'tender', ',', 'traders', 'said', '.']
['Rebates', 'requested', 'ranged', 'from', '127.75', 'to', '132.50', 'European', 'Currency', 'Units', 'a', 'tonne', 'for', 'maize', ',', '136.00', 'to',
'141.00', 'Ecus', 'a', 'tonne', 'for', 'barley', 'and', '134.25', 'to', '141.81', 'Ecus', 'for', 'bread', 'wheat', ',', 'while', 'rebates', 'requested', 'for',
'feed', 'wheat', 'were', '137.65', 'Ecus', ',', 'they', 'said', '.']
```

Stanford Parser

- The Stanford Parser is a probabilistic natural language parser written in Java.
- It can be used for most of the tasks discussed before.
- There exist APIs in many other languages.
  - https://stanfordnlp.github.io/CoreNLP/other-languages.html
- A server implementation can be downloaded and used to distribute NLP tasks to a server.
  - https://stanfordnlp.github.io/CoreNLP/corenlp-server.html

Stanford Parser – Maven

```
<dependencies>
    <dependency>
        <groupId>edu.stanford.nlp</groupId>
        <artifactId>stanford-corenlp</artifactId>
        <version>3.9.2</version>
    </dependency>
    <dependency>
        <groupId>edu.stanford.nlp</groupId>
        <artifactId>stanford-corenlp</artifactId>
        <version>3.9.2</version>
        <classifier>models</classifier>
    </dependency>
</dependencies>
```

Tokenization - Stanford Parser

```
Properties props = new Properties();
props.setProperty("annotators", "tokenize, ssplit, pos, lemma");
StanfordCoreNLP pipeline = new StanfordCoreNLP(props);
String text = "She went to America last week.";
Annotation document = new Annotation(text);
pipeline.annotate(document);

List<CoreMap> sentences = document.get(CoreAnnotations.SentencesAnnotation.class);
for (CoreMap sentence : sentences) {
  List<CoreLabel> tokens = sentence.get(CoreAnnotations.TokensAnnotation.class);
  tokens.forEach(t -> System.out.println(t.word()));
  tokens.forEach(t -> System.out.println(t.lemma()));
}
```

Stemming & Stopwords - Stanford Parser

- **Stemming** reduces word-forms to (pseudo)stems.
- **Lemmatization** reduces word-forms to linguistically valid lemmas.
- **Stopwords** are words that add no semantic value. They might have an impact on similarity values and topic models unless they are filtered using stop word lists.
  - There is no native annotator for stop words in Stanford Parser. A list of stopwords can be found at: https://github.com/stanfordnlp/CoreNLP/blob/master/data/edu/stanford/nlp/patterns/surface/stopwords.txt.

Natural Language Processing (NLP)

… stopword filtering and stemming with Porter Stemmer

In [39]:
```
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords

stemmer = PorterStemmer()
for sentence in sentences:
    print([stemmer.stem(x) for x in word_tokenize(sentence)
           if x not in stopwords.words('english')])
```

```
['french', 'free', 'market', 'cereal', 'export', 'bid', 'detail', 'french', 'oper', 'request', 'licenc', 'export', '675,500', 'tonn', 'maiz', ',', '245,000',
'tonn', 'barley', ',', '22,000', 'tonn', 'soft', 'bread', 'wheat', '20,000', 'tonn', 'feed', 'wheat', 'today', "'s", 'european', 'commun', 'tender', ',',
'trader', 'said', '.']
['rebat', 'request', 'rang', '127.75', '132.50', 'european', 'currenc', 'unit', 'tonn', 'maiz', ',', '136.00', '141.00', 'ecu', 'tonn', 'barley', '134.25',
'141.81', 'ecu', 'bread', 'wheat', ',', 'rebat', 'request', 'feed', 'wheat', '137.65', 'ecu', ',', 'said', '.']
```

Topic Models

https://en.wikipedia.org/wiki/Topic_model

- Topic Models provide a first overview on document collections.
- "dog" and "bone" are words in the same topic. "cat" and "meow" are in another. "the" and "is" will appear equally.
- One document may have multiple topics.
- Topic Modeling produces clusters of similar words.

NLP with Topic Models

Read a natural language corpus and preprocess it.

In [*]:

```
import re

# Final processing pipeline.
def preprocess(text):
    result = []
    for sentence in sent_tokenize(text, language='english'):
        for x in [stemmer.stem(x) for x in word_tokenize(sentence)
                  if x not in stopwords.words('english')]:
            result.append(re.sub(r"[^A-Za-z]+", '', x))

    return result


documents = [preprocess(reuters.raw(x)) for x in reuters.fileids()[:200]]
```

NLP with Topic Models

Use the ‘gensim’ package and apply LDA to recover topics of this corpus.

In [299]:
```
from gensim.corpora.dictionary import Dictionary
from gensim import models

dictionary = Dictionary(documents)
dictionary.filter_extremes(no_below=3, no_above=0.3)
corpus = [dictionary.doc2bow(x) for x in documents]

# Train the model on the corpus.
lda = models.LdaModel(corpus, num_topics=20, alpha='auto', id2word=dictionary)

for i in range(0, lda.num_topics-1):
   print(lda.print_topic(i, topn=7))
```

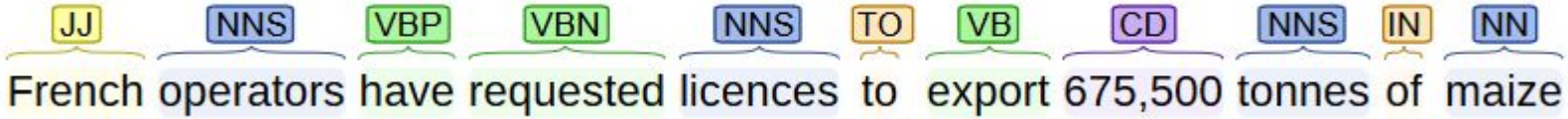
```
0.028*"vs" + 0.021*"may" + 0.021*"us" + 0.020*"trade" + 0.014*"div" + 0.014*"record" + 0.013*"prior"
0.022*"bank" + 0.021*"stg" + 0.015*"us" + 0.015*"pct" + 0.014*"compani" + 0.013*"market" + 0.010*"first"
0.016*"inc" + 0.015*"japan" + 0.014*"share" + 0.012*"compani" + 0.011*"pct" + 0.011*"trade" + 0.010*"stock"
0.031*"vs" + 0.015*"pct" + 0.014*"would" + 0.013*"shr" + 0.012*"china" + 0.011*"qtr" + 0.011*"compani"
0.018*"would" + 0.016*"share" + 0.012*"bank" + 0.010*"pct" + 0.009*"us" + 0.009*"report" + 0.008*"trade"
0.031*"billion" + 0.015*"trade" + 0.012*"pct" + 0.011*"report" + 0.011*"bank" + 0.010*"mark" + 0.009*"compani"
0.019*"compani" + 0.013*"pct" + 0.012*"us" + 0.011*"market" + 0.009*"rate" + 0.009*"earn" + 0.009*"trade"
```

Topic Models

- Stanford Parser supports topic modelling in a separate component. https://nlp.stanford.edu/software/tmt/tmt-0.4/
- Spark Machine Learning Library (Mlib) supports topic modeling.
  - https://medium.com/zero-gravity-labs/lda-topic-modeling-in-spark-mllib-febe84b9432

Parsing Natural Language

Stanford Parser can be used for more complex NLP tasks. Words are tagged on whether they are nouns, verbs or adjectives, etc.




```
#Retrieve all nouns:
from nltk.parse.corenlp import CoreNLPDependencyParser
dep_parser = CoreNLPDependencyParser(url='http://localhost:9000')
parse, = dep_parser.raw_parse(text)

nouns = []
for node, nodedict in parse.nodes.items():
    if 'NN' in nodedict['tag']:
        nouns.append(nodedict['word'])
```

Part-of-speech - Stanford Parser in Java

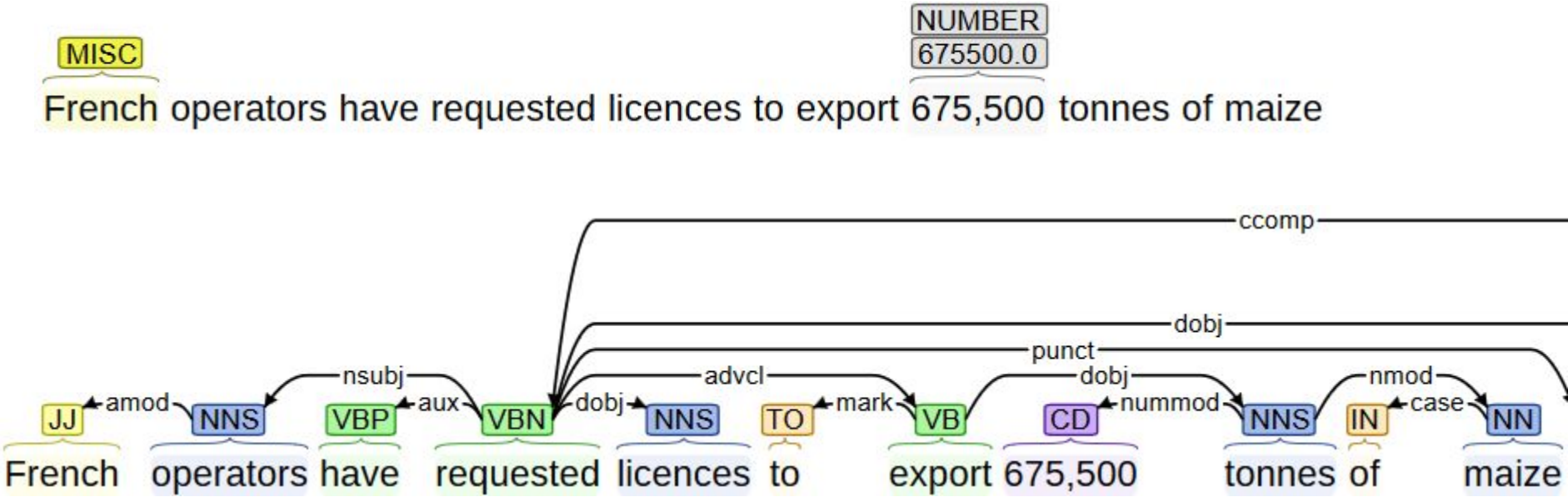
See Gist: https://gist.github.com/MarcelH91/903c01a8b5a1a51f9ab345bde3de7bf7

Stanford CoreNLP Server

- Start Server
- Open your webbrowser and access localhost:9000

Parsing Natural Language

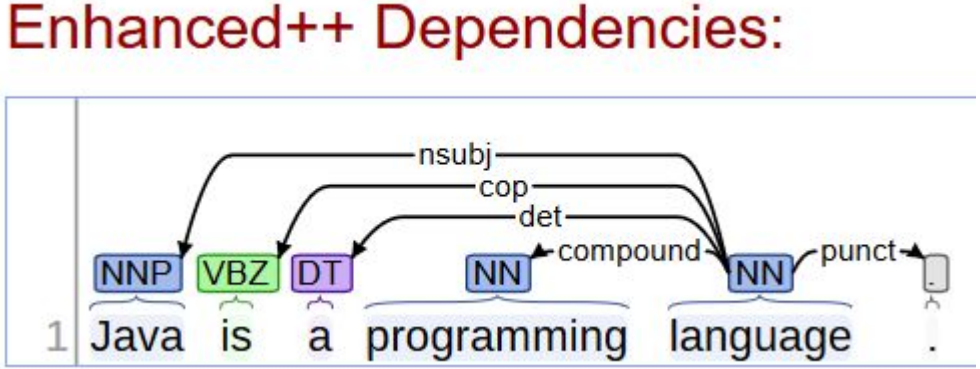
Named entity recognition and sentence structure analysis are supported by Stanford Parser.



Dependency Parsing - Stanford Parser

- Start Server
- Open your webbrowser and access localhost:9000

- Dependency Parsing implies a grammar-based analysis of sentence structure. Dependencies between words are recovered.
- For Stanforder, an overview on dependency names is provided at: https://nlp.stanford.edu/software/dependencies_manual.pdf



Dependency Parsing - Semgrex

- Semgrexes are regular expressions on parsed sentences including dependencies.
  - Matching a hypernym after ‘is a’:

{pos:/NN./}=hypernym >cop {pos:/VB.*/}=Verb >det {}=det*

  - Matching part-of relationship after ‘is one of’:

{pos:CD;word:one}=one >cop {pos:/VB./}=Verb >=rel {pos:/NN.*/}=hypernym*

Summary

- Tokenization is an important first step.
- Stemming can be used to reduce complexity and raise similarity.
- Stop words can be removed in preprocessing.
- Based on word vectors TF-IDF can be computed.
- Topic Models provide overviews.
- Similarity measures can be used to retrieve clusters.
- Part-of-speech analysis allows precise analysis.
- Dependency analysis allows more precise analysis.