Intro to NLP

with Udpipe and Spacy

Session # 4

TABA @ CBA Batch 12

Sudhir Voleti

1

Session Plan

- Intro to Elementary NLP
  - Annotations, POS tagging, NER, Chunking, etc.
- NLP in Py with Spacy
  - Spacy explorations
- NLP in R with Udpipe
  - UDPipe explorations
- NLP in other Languages
  - Hindi, Spanish etc.

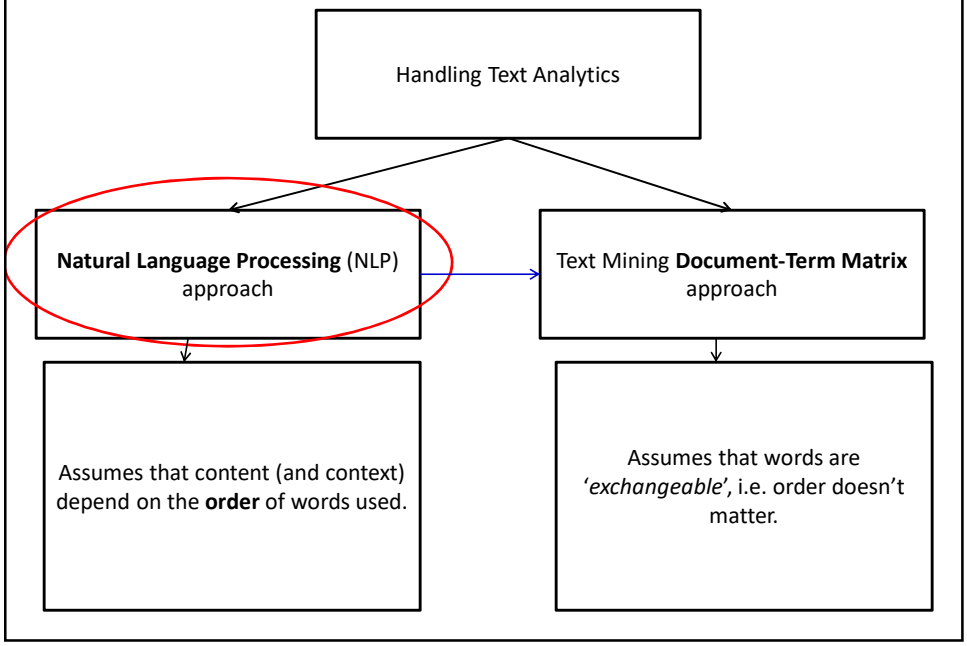
2

2

Elementary NLP

3

Definitional Preliminaries: NLP Vs Bag-of-words



4

Definitional Preliminaries: On NLP

- What is NLP?

  Natural language processing (NLP) is a set of techniques for using computers to detect in human language the kinds of things that humans detect *automatically*.

  'Kinds of things' here imply entities, relationships, context, meaning ...

- What are some things we humans process automatically when reading or writing 'natural' text?

  [1] We automatically *parse* a text into structural units – paras, sentences.
  [2] We implicitly recognize *parts-of-speech* (nouns, verbs etc.).
  [3] We recognize people, places, dates etc. ('*entities*') as they come up.
  [4] We process subject, object, tense, count etc. automatically
  [5] And we judge whether text is happy or sad etc., automatically.
  [6] Etc.

5

Requirements of an NLP Workflow

- An *ideal* NLP workflow should be able to do:

- [1] Tokenisation
- [2] Parts of speech tagging (POSTagging)
- [3] Lemmatisation & Stemming
- [4] Morphological feature tagging
- [5] Syntactic dependency parsing
- [6] Named entity recognition (NER)
- [7] Extracting word & sentence meaning …

- One can evaluate the functionalities of different NLP libraries against this list.

- We today cover Py's Spacy and R's UDPipe.

6

Example Demo 1: Phrase-Parsing a Sentence

- "Donald Trump is a controversial American President."
- What *phrases* can you ID in the above sentence?

- Tagging it for **Parts-of-speech (POS)** yields this:

```
> ## Demo 1
> sent1 = "Donald Trump is a controversial American President."
> ann_sent1 = sent1 %>% py.annotate(); ann_sent1
  pos_tag slnum         token doc_num                        Description
4     NNP     1        Donald       1                Proper noun, singular
5     NNP     2         Trump       1                Proper noun, singular
7     VBZ     3            is       1 Verb, 3rd person singular present
1      DT     4             a       1                          Determiner
2      JJ     5 controversial       1                           Adjective
3      JJ     6      American       1                           Adjective
6     NNP     7     President       1                Proper noun, singular
>
>
> ann_sent1 %>% extract_phrases(noun=TRUE)
$`1`
[1] "Donald Trump"                    "controversial American President"
```

7

Example Demo 1a: Parsing Verb Phrases

- "The Rover separated cleanly and landed perfectly on Mars."

- What verb phrases (VPs) do you detect there? Based on what pattern?

```
> ann_sent2 <- sent2 %>% py.annotate();  ann_sent2
  pos_tag slnum     token doc_num                                Description
2      DT     1       The       1                                Determiner
5     NNP     2     Rover       1                Proper noun, singular
8     VBD     3 separated       1                     Verb, past tense
6      RB     4   cleanly       1                              Adverb
1      CC     5       and       1          Coordinating conjunction
9     VBD     6    landed       1                     Verb, past tense
7      RB     7 perfectly       1                              Adverb
3      IN     8        on       1 Preposition or subordinating conjunction
4     NNP     9      Mars       1                Proper noun, singular

> sent2 %>% py.annotate() %>% extract_phrases(noun=FALSE)
$`1`
[1] "separated cleanly" "landed perfectly"
```

8

Example Demo 2: Syntactic Dependency Parsing

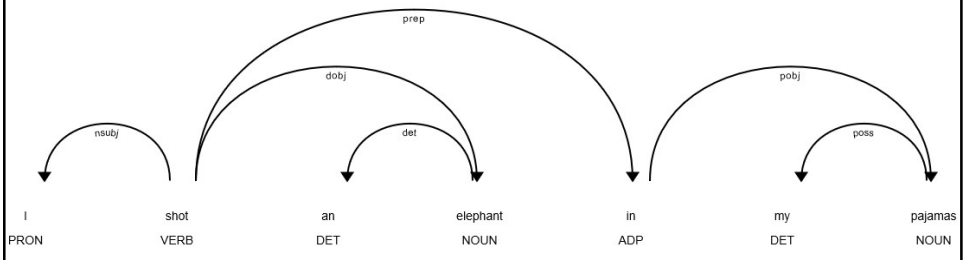
- Finding which POS_Objects **depend** on which other POSO *syntactically*.
- "I prefer the morning flight through Denver."
- Qs to consider:
- Who/What is the *SUBJECT*? **%>%** What does the subject *do* (ROOT verb)? **%>%** What (noun) *OBJECTS* do those actions *affect*? %>% Etc.

```
> s %>% cnlp_annotate(., as_strings = TRUE) %>%
+     cnlp_get_dependency(., get_token = TRUE)
# A tibble: 8 x 10
  id      sid   tid tid_target relation relation_full word  lemma word_target
  <chr> <int> <int>      <int> <chr>    <chr>         <chr> <chr> <chr>
1 doc1      1     2          1 nsubj    nsubj         pref~ pref~ I
2 doc1      1     0          2 root     root          ROOT  ROOT  prefer
3 doc1      1     5          3 det      det           flig~ flig~ the
4 doc1      1     5          4 compound compound      flig~ flig~ morning
5 doc1      1     2          5 obj      obj           pref~ pref~ flight
6 doc1      1     7          6 case     case          Denv~ Denv~ through
7 doc1      1     5          7 nmod     nmod          flig~ flig~ Denver
8 doc1      1     2          8 punct    punct         pref~ pref~ .
# ... with 1 more variable: lemma_target <chr>
```

9

Syntactic Dependency parsing: Example

- Here's an example from spacy to illustrate some NLP functionalities.
- Sentence: "I shot an elephant in my pajamas."
- Punchline: "Wonder how it got into my pajamas in the first place…"



10

Example Demo 3: Named Entity Recognition (NER)

- Named entities such as Persons, Organizations, Geopolitical Entities or GPEs are the most common use-cases.
- Mining the annotated Trump sentence for NER yields this:

```
> ner_sent1 <- sent1 %>% py.ner(); ner_sent1
  slnum          token pos_tag   named_entity
1     1         Donald     NNP       B-PERSON
2     2          Trump     NNP B-ORGANIZATION
3     3             is     VBZ                O
4     4              a      DT                O
5     5 controversial      JJ                O
6     6       American      JJ          B-GPE
7     7      President     NNP                O
8     8              .       .                O
```

11

Intro to NLP: Recap

- What is NLP?
- What main functionalities come in NLP?
- What are the advantages and disadvantages of NLP over BOW (for insight extraction)?

12

12

Session Plan

- Intro to Elementary NLP
  - Annotations, POS tagging, NER, Chunking, etc.
- NLP in Py with Spacy
  - Spacy explorations
- NLP in R with Udpipe
  - UDPipe explorations
- NLP in other Languages
  - Hindi, Spanish etc.

13

13

NLP in Py with *Spacy*

14

Some Simple NLP Ops with Spacy in Py

- Let's perform some simple NLP operations in Py.
- Open file '*Spacy explorations.ipynb*'
- Now answer these Qs:
- [1] What functionality did we just cover in NLP?
- [2] What py modules were imported? What funcs were called?
- [3] What functionality was left over?

15

Session Plan

- Intro to Elementary NLP
  - Annotations, POS tagging, NER, Chunking, etc.
- NLP in Py with Spacy
  - Spacy explorations
- NLP in R with Udpipe
  - UDPipe explorations
- NLP in other Languages
  - Hindi, Spanish etc.

16

16

NLP in R with *UDpipe*

17

Introducing UDPipe

- Till recently, R had great BoW capabilities but weak NLP ones. Then UDPipe changed that quite a bit.
- Recall also that so far in what we have done we are yet to see 2 NLP requirements pour through, namely:
  - Phrase extractions
  - Morphological feature tagging
- Let's see these two laid out as well in a pure R package which offers trained models in as many as 52 languages!
- Open file '*udpipe explorations.Rmd*'

18

18

NLP with Udpipe - Recap

- So how did you find UDpipe?
- What major NLP ops did we see?
- The big advantage of having the entire workflow within R sans external dependencies is ….

19

Session Plan

- Intro to Elementary NLP
  - Annotations, POS tagging, NER, Chunking, etc.
- NLP in Py with Spacy
  - Spacy explorations
- NLP in R with Udpipe
  - UDPipe explorations

- NLP in other Languages
  - Hindi, Spanish etc.

20

20

NLP in other Languages

Introducing UDPipe

21

UDPipe Recap

- Now that we have covered UDPipe, answer these basic Qs:
- What main libraries and modules did we call?
- What main functions did we see?
- Which among the NLP requirements have we not met yet?
- When might you use udpipe and when not?

22

Q & A

23