

# Solution guideline

Katarzyna Sornat

# 1. Reading the data

Below we have an example of input data. Even though the order of the columns was suggested, we clearly see (yellow highlighted) that we cannot rely on this because:

- The order of the columns can be different
- We do not have a clear separator information

```
id,title,authors,venue,year
0,semantic integration of environmental models for application to global information systems and decision-making d. scott mackay,,sigmod record,1999.0
1,estimation of query-result distribution and its application in parallel-join load balancing vldb 1996,"viswanath poosala , yannis e. ioannidis",,
2,incremental maintenance for non-distributive aggregate functions vldb 2002,"themistoklis palpanas , richard sidle , hamid pirahesh , roberta cochrane",,
3,"cost-based selection of path expression processing algorithms in object-oriented databases zhao-hui tang , georges gardarin , jean-robert gruser vldb 1996",,,
4,"benchmarking spatial join operations with spatial output erik g. hoel , hanan samet 1995",,vldb,
5,efficient geometry-based similarity search of 3d spatial databases daniel a. keim sigmod conference,,,1999.0
6,mining the world wide web : an information search approach - book review aris m. ouksel sigmod record 2002,,,
_ _ _ _ _
```

That was the reason why I read everything line by line without any splitting.

# 1. Reading the data

Easy to do was to take an id of the document in each .csv file, because I noticed that it should be first integer occurring in the text – id column (here after splitting). The rest of the text – after removing this first integer, was untouched as an input for further transformation (column description).

id	description
0	semantic integration of environmental models for application to global informatio...
1	estimation of query-result distribution and its application in parallel-join load bala...
2	incremental maintenance for non-distributive aggregate functions vldb 2002,"the...
3	"cost-based selection of path expression processing algorithms in object-oriented...
4	"benchmarking spatial join operations with spatial output erik g. hoel , hanan sam...
5	efficient geometry-based similarity search of 3d spatial databases daniel a. keim si...
6	mining the world wide web : an information search approach - book review aris m...
7	enhanced abstract data types in object-relational databases vldb j. 1998,praveen s...
8	"report on dart ' 96 : databases : active and real-time ( concepts meet practice ) na...
9	unisql 's next-generation object-relational database management system sigmod ...
10	dual-buffering strategies in object bases 1994,"alfons kemper , donald kossmann",...

## 2. Data cleaning & transformation

Cleaning of the data and creating the features was done in R as well as in Python:

- Removing digits
- Removing punctuation
- Removing whitespaces (and also trim the one at the beginning and by the end)
- Removing words where `len(word) <= 2`

### Before cleaning

```
[1] "semantic integration of environmental models for application to global information systems and decision-making d. scott mackay,,sigmod record,1999"
```

### After cleaning


```
"semantic integration environmental models for application global information systems and decision making scott mackay sigmod record"
```

## 2. Data cleaning & transformation

As we have a plain text, three important informations are worth to retrieve:

- A year of the publication
- A Venue
- The authors

I managed to get the year of the publication for most of the rows. It required some cleaning as the year could also be a double number, and that was not grabbed by digit regular expression. An example below:

 tableA.csv — Notatnik

Plik Edycja Format Widok Pomoc

id,title,authors,venue,year

0,semantic integration of environmental models for application to global information systems and decision-making d. scott mackay,,sigmod record,1999.0

### 3. Fuzzywuzzy & SpaCy for feature creation

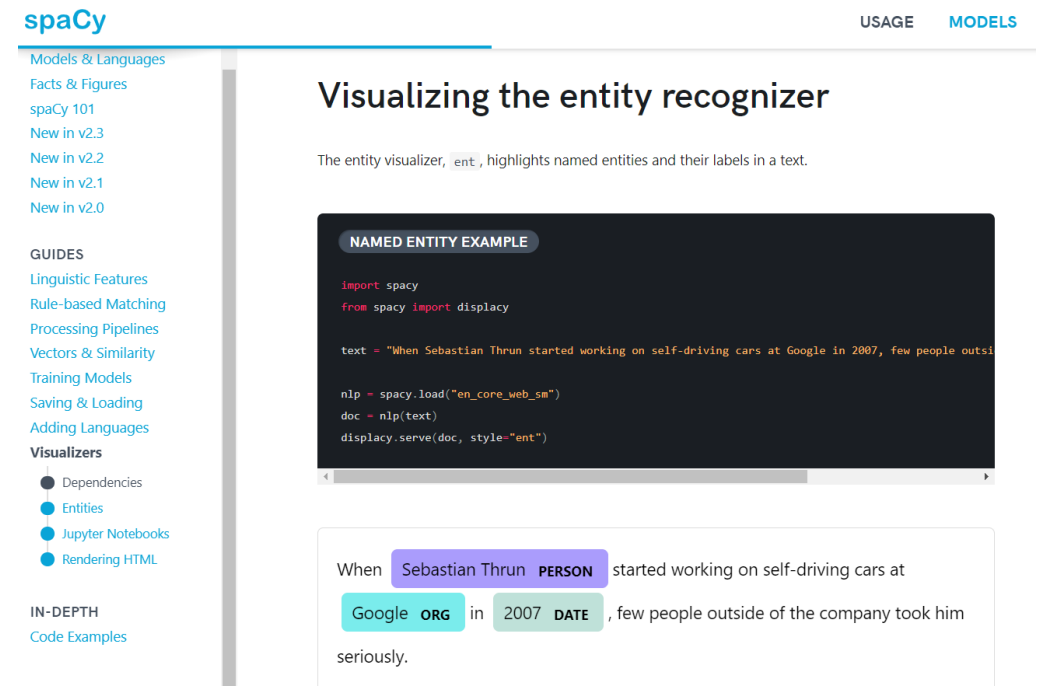
As the time was limited, I decided to use fuzzywuzzy package from Python to calculate various definitions of distances provided by the package.

Features created here were for example:

- Size of token intersection between two
- Similarity measure between ordered vector of tokens for each pair

All the details can be found in the notebook fuzzywuzzy\_features.ipynb

Remark: I think that retrieving authors would be very helpful but I did not manage to do this. For the next steps my approach would be to use:



The screenshot shows the spaCy website with a sidebar on the left containing links like 'Models & Languages', 'Facts & Figures', 'spaCy 101', 'New in v2.3', 'New in v2.2', 'New in v2.1', 'New in v2.0', 'GUIDES', 'Linguistic Features', 'Rule-based Matching', 'Processing Pipelines', 'Vectors & Similarity', 'Training Models', 'Saving & Loading', 'Adding Languages', 'Visualizers' (with sub-links for Dependencies, Entities, Jupyter Notebooks, and Rendering HTML), 'IN-DEPTH', and 'Code Examples'. The main content area is titled 'Visualizing the entity recognizer' and includes a sub-header 'NAMED ENTITY EXAMPLE'. Below this is a code block showing Python code for loading a spaCy model and visualizing a document. The code is: 

```
import spacy
from spacy import displacy

text = "When Sebastian Thrun started working on self-driving cars at Google in 2007, few people outside of the company took him seriously."

nlp = spacy.load("en_core_web_sm")
doc = nlp(text)
displacy.serve(doc, style="ent")
```

 Below the code is a visual representation of the text where entities are highlighted: 'Sebastian Thrun' is labeled 'PERSON', 'Google' is labeled 'ORG', and '2007' is labeled 'DATE'. The text is: 'When Sebastian Thrun PERSON started working on self-driving cars at Google ORG in 2007 DATE, few people outside of the company took him seriously.'

### 3. Fuzzywuzzy & SpaCy for feature creation

That approach gave me some bunch of features based mostly on the distance between two sentences.

Filter										Q
label	fuzzy_similarity_sorted_tokens_perc	fuzzy_similarity_partial_ratio_perc	len_tokens_x	len_tokens_y	len_common_tokens	common_to_smaller_ratio	common_to_longer_ratio	fuzzy_similarity_perc	if_years_eq	
0	0.47	0.45	22	18	4	0.2222222	0.18181818	0.46		
0	0.54	0.62	6	7	2	0.3333333	0.28571429	0.56		
0	0.43	0.45	17	12	2	0.1666667	0.11764706	0.43		
1	0.87	0.78	18	22	17	0.9444444	0.77272727	0.71		
0	0.54	0.50	19	17	4	0.2352941	0.21052632	0.49		
1	0.91	0.78	17	18	15	0.8823529	0.83333333	0.83		
0	0.36	0.63	8	21	3	0.3750000	0.14285714	0.40		
0	0.56	0.67	7	19	6	0.8571429	0.31578947	0.45		
0	0.50	0.49	14	16	3	0.2142857	0.18750000	0.47		
0	0.55	0.55	15	20	6	0.4000000	0.30000000	0.56		
0	0.49	0.54	10	26	3	0.3000000	0.11538462	0.47		
1	0.89	0.80	14	16	13	0.9285714	0.81250000	0.76		

### 3. Final approach

I picked logistic regression as a first shoot. The cutoff I picked was 0.4. As the data were not so much balanced ( $\sim 0.8$  – class 0) I would manipulate a bit more with undersampling & weights to secure that it will not overfit.

B