1.NER regex. It contains the main script "ner_regex.py" and the "Dataset" folder. The Dataset folder is provided with an example "soa.csv" input file containing documents.
Usage: The user can just execute the script ner_regex.py. This module is meant to find text-instances of SOA data and to label them.
Output: a .json1 file containing annotations that describe the retrieved entities.

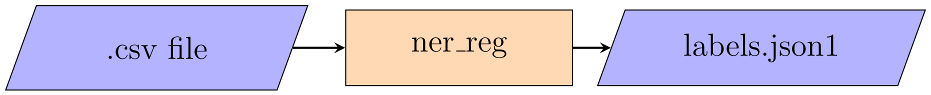


Figure 1. Module 1

2.Dataset splitter.
Input: The gold.json1 file is an example of **Ground Truth**. This module takes the gold.json1 and divides creates the docbins that will be used later on for building a NN.
Usage: Execute the dataset_splitter.py script
Output: Train.spacy, Dev.spacy, Test.spacy.

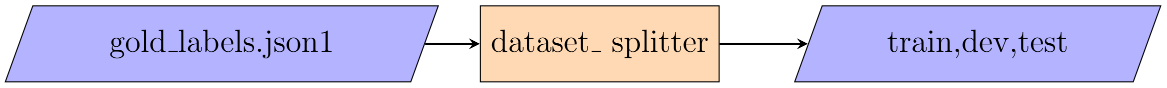


Figure 2. Module 2

3.NER spaCy.
Inputs: the module needs the docbins "train.spacy","dev.spacy","test.spacy" produced by the Dataset Splitter Module.
Usage:

(1) execute script 3.0-retrieve-inputs.sh, it will copy the spacy file from the previous module
(2) 3.1-train.sh: trains a neural network
(3) 3.2-debug.sh: retrieves info on how experienced the NN is.
(4) 3.3-evaluate.sh: tests the NN with the test data of the test.spacy file
(5) 3.4-usage.py

Outputs: NN model with debug information and testing results.

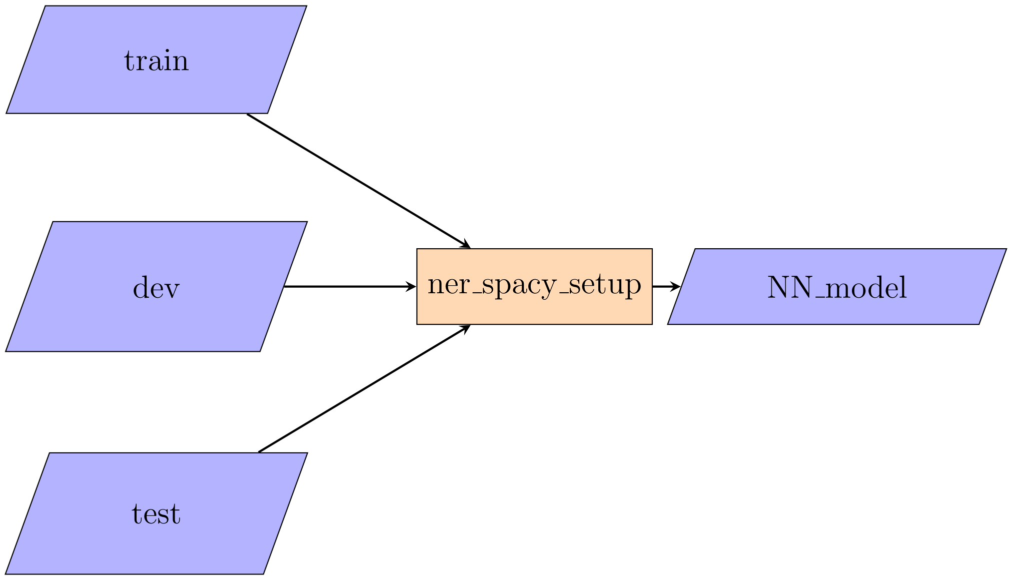


FIGURE 3. Setup of the NN

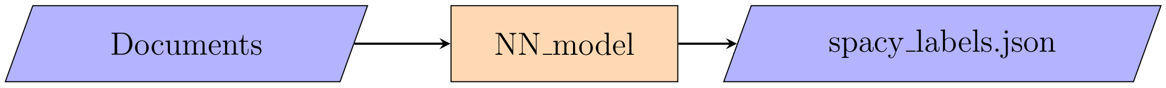
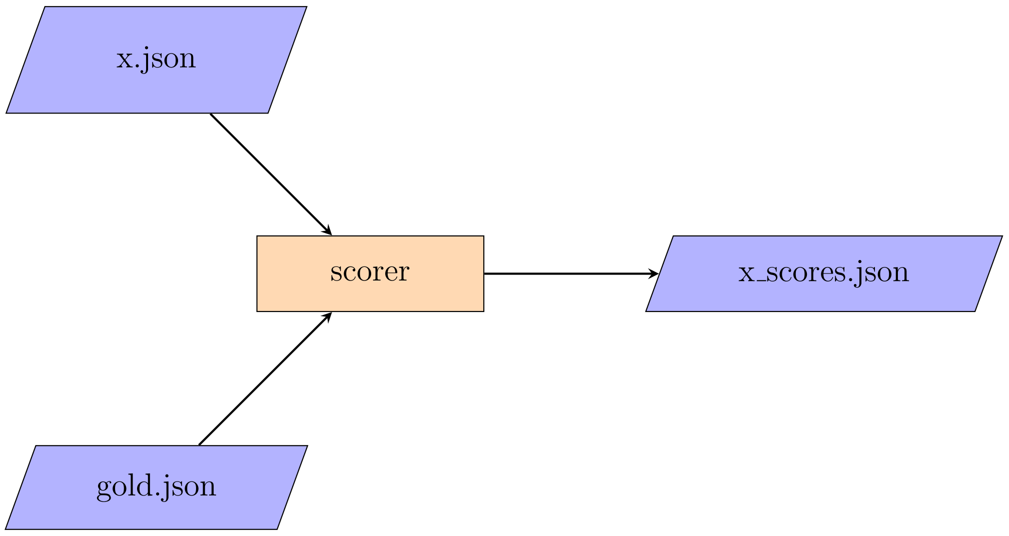
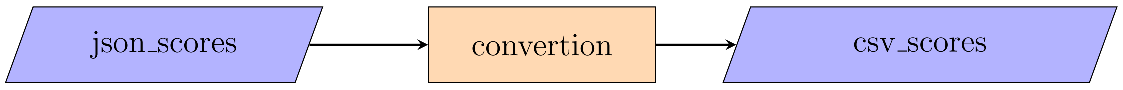


FIGURE 4. Usage of the NN

Scorer.
Inputs: gold.json1, describes the Ground Truth
Usage: 1. 4.0-retrieve-inputs.sh: retrieves the ner.json1 produced by the NER 2. scorer.py: scores the ner.json1 by comparing it to the gold.json
Outputs: output-cat-class.json: scores for every category item and classification item



Json to csv.
Usage: put here the score file derived from the scorer output
Run: json_to_csv.py
Output: csv_tables/ner_results.csv, describes the scores in a tabular form


json_scores
convertion
csv_scores