Community Workshop

Practical Introduction to Spacy Pattern Matching

Friederike Bauer

Who we are

We are a Europe-wide network of over 2,400 data enthusiasts who want to improve the world through and with data.

#MetaWorldSavior

What we do

Projects

We carry out pro bono data analysis projects for non-profit organizations, enabling civil society to work in an evidence-based and efficient manner.

Education

We offer socially engaged data analysts and data-interested and non-profit organizations opportunities to improve their knowledge about data.

Community

We connect socially engaged data enthusiasts and enter into a dialog about the value and benefits of data and data analysis for the common good.

What will we do?

1) Main goals
2) Zooming out: why pattern matching?
3) Rule-based matching overview
4) Spacy vocabulary
5) Hands on

…

6) Considerations
7) My Learnings

Main Goals

In this course we will look into rule-based matching with spaCy. After a short theoretical introduction to pattern matching and token attributes we will dive into some practical expercises. It is advised to have some python knowledge

Main Goals

In this course we will look into rule-based matching with spaCy. After a short theoretical introduction to pattern matching and token attributes we will dive into some practical expercises. It is advised to have some python knowledge

Quick start to Spacy pattern matching + getting excited about it ☺

Who am I?

- Friederike Bauer
- Data Scientist / Software Developer (Frontend) @and effect
- MSc Social and Economic Data Science (SEDS) (@Konstanz)
- BA Social Science (@Stuttgart)
- Projects with spacy + pattern matching

Zooming out: why pattern matching?

- Part of NLP (natural language processing)
- Can be done with statistical models or rule-based approaches
- Used as a step in many analysis pipelines, like…

Use Cases for Pattern Matching

- Auto-correct or auto-complete processes

- Information retrieval

- Language translation

- Text classification* (our use-case today)

… many more

* Text classification = assign text to pre-defined classes (two or more)

Rule based matching overview (in spaCy)

Model / Matching-Type	How does it match?	Use Case
Token Matcher	Token-based matching (on token attributes)	Searching for sequences based on lemmas, POS tags, etc.
Phrase Matcher	Allows to use large terminology lists (and create doc objects)	Same as Token Matcher but for larger terminology lists
Dependency Matcher	"the DependencyMatcher patterns match tokens in the dependency parse and specify the relations between them." [1]	If you want to focus on dependencies between phrases
Entity Ruler	"lets you add named entities based on pattern dictionaries." [2]	Pipeline component, if you are also interested in entities. A token can only belong to one class
Span Ruler	"generalized version of the entity ruler that lets you add spans to doc.spans or doc.ents based on pattern dictionaries[3]	Same span patterns as entity ruler. A token can belong to multiple classes

[1] https://spacy.io/usage/rule-based-matching#dependencymatcher

[2] https://spacy.io/usage/rule-based-matching#entityruler

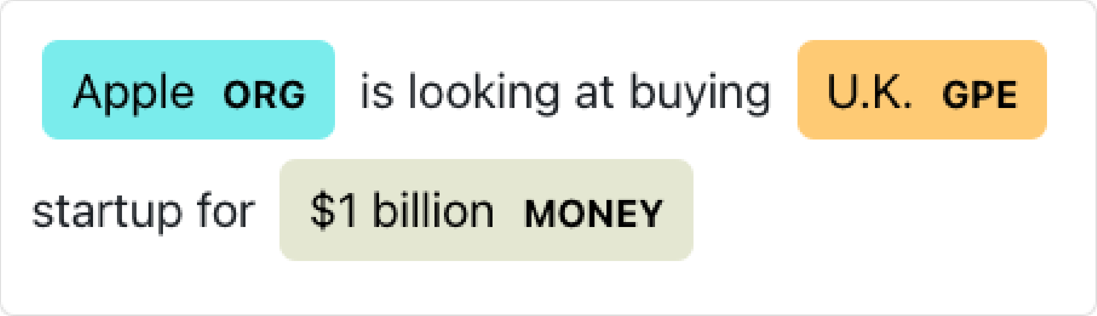
[3] https://spacy.io/usage/rule-based-matching#spanruler

I have found this video helpful to understand the differences: https://www.youtube.com/watch?v=4vZoAg90mtI

SpaCy vocabulary: entity

Entity

= “A named entity is a “real-world object” that’s assigned a name – for example, a person, a country, a product or a book title. spaCy can recognize various types of named entities in a document, by asking the model for a prediction.”[1]



Apple ORG is looking at buying U.K. GPE startup for $1 billion MONEY

Also see Spacy Linguistic Features Intro: https://spacy.io/usage/linguistic-features

[1] https://spacy.io/usage/linguistic-features#named-entities

SpaCy vocabulary: token

Token

= individual unit of text (split from a larger sequence like a sentence)

For example:

- A word
- A number
- Punctuation mark

Text = “Apple is looking at buying U.K. startup for $1 billion”

Tokens = [“Apple”, “is”, “looking”,…]

Tokenization

= process of splitting raw text into tokens (using a tokenizer)

SpaCy vocabulary: token attributes

Token attributes

= spacy stores information for each token, like the normalized form of the token or the base form (lemma). It also checks for example if a token is a number or if it starts the sentence….

See: https://spacy.io/api/token#attributes

SpaCy vocabularity: phrase

Phrase

= multi-word expression or group of tokens
= sequence of words
(= sentence or multiple words)

For example: “Apple is looking at buying U.K. startup for $1 billion”

Also see Spacy Linguistic Features Intro: https://spacy.io/usage/linguistic-features

Hands on

- Jupyter Notebook is provided
- Sample Dataset through Kaggle
- Sample Patterns

Option 1)
GitHub Repo
→ https://github.com/CorrelAid/workshop-spacy

Option 2)
https://colab.research.google.com/

Option 3)
Watch me run the code ☺

Considerations

→ How much cleaning of the text do you want to do?

For example:

- lowercase?
- Normalize? Remove punctuation?
- Remove other elements like emojis in text?
- Converting dates?

→ Will it help matching? Or will it lead to false matching?

Learnings: pros and cons

+ Useful for high precision

+ tailoring to a specific domain

- A lot of manual work and testing
- Low recall
- Ambiguity (words depend on context + have different meanings)

Learnings: each language is different

→ Make sure you take into account the specifics of the language you are creating the patterns for

For example - things to consider when using german:

- Äüö can also be written ae ue oe
- ß can also be written ss
- Plurals are often created similarly
- Sie/ du

Learnings: Testing

- Some concepts might be very closely related → calculate co-occurrence matrix to see for example
- Write tests to exclude phrases from one group from the other
- Write tests to check if your tokens are created correctly (there are different tokenizers…)
- Look into what you did not find → why does your pattern not catch certain tokens/ phrases
- Look into \n (line break) and trailing white space → they do not show up when printing ☺ also check for correct UTF encoding (probably UTF-8)
- Use a logger to check your pattern generation + tests

Additional Ressources

- Glossary of linguistic terms: https://glossary.sil.org/term
- Spacy rule-based matcher explorer: https://demos.explosion.ai/matcher
- Spacy Display Playground: https://demos.explosion.ai/displacy
- Regex Playground/ Explorer: https://regex101.com/
- … links for specific use-cases in the practical Jupyter Notebook

Open Questions ?

Your feedback – thank you 🤍

https://ee.correlaid.org/single/J5XjABxW?return_url=https://www.correlaid.org

Upcoming Workshops

Community Workshops

🏴󠁧󠁢󠁥󠁮󠁧󠁿 for data scientists and data interested

Git for Newbies

Oct 21th, 19:00 - 20:00

How to: Vorstand & Ethikkommission

Nov 5th, 18:00 - 19:00

Git for Newbies

Dec 3rd, 18:00 - 19:00

CorrelCompact

🇩🇪 for data world beginners

KI-Kickstart - Grundlagen und Chancen für Non-Profits

Oct 22th, 17:00 - 17:45

Themenabend Generative KI

Oct 30th, 18:30 - 19:30

Mission Datenqualität - vom Rohmaterial zum Datengold

Nov 5th, 14:00 - 14:45