6. How to use the spaCy Matcher

Dr. W.J.B. Mattingly
Smithsonian Data Science Lab and United States Holocaust Memorial Museum
August 2021


```
import spacy
```

```
from spacy.matcher import Matcher
```

6.1. Basic Example

```
nlp = spacy.load("en_core_web_sm")
matcher = Matcher(nlp.vocab)
pattern = [{"LIKE_EMAIL": True}]
matcher.add("EMAIL_ADDRESS", [pattern])
doc = nlp("This is an email address: wmattingly@aol.com")
matches = matcher(doc)
```

```
print (matches)
```

```
[(16571425990740197027, 6, 7)]
```

Lexeme, start token, end token

```
print (nlp.vocab[matches[0][0]].text)
```

```
EMAIL_ADDRESS
```

6.2. Attributes Taken by Matcher

- ORTH - The exact verbatim of a token (str)
- TEXT - The exact verbatim of a token (str)
- LOWER - The lowercase form of the token text (str)
- LENGTH - The length of the token text (int)
- IS_ALPHA
- IS_ASCII
- IS_DIGIT
- IS_LOWER
- IS_UPPER
- IS_TITLE
- IS_PUNCT
- IS_SPACE
- IS_STOP
- IS_SENT_START
- LIKE_NUM
- LIKE_URL
- LIKE_EMAIL
- SPACY
- POS
- TAG
- MORPH
- DEP
- LEMMA
- SHAPE
- ENT_TYPE

- _ - Custom extension attributes (Dict[str, Any])
- OP

6.3. Applied Matcher

```
with open ("data/wiki_mlk.txt", "r") as f:
    text = f.read()
```

print (text)

```
nlp = spacy.load("en_core_web_sm")
```

6.4. Grabbing all Proper Nouns

```
matcher = Matcher(nlp.vocab)
pattern = [{"POS": "PROPN"}]
matcher.add("PROPER_NOUNS", [pattern])
doc = nlp(text)
matches = matcher(doc)
print (len(matches))
for match in matches[:10]:
    print (match, doc[match[1]:match[2]])
```

```
102
(3232560085755078826, 0, 1) Martin
(3232560085755078826, 1, 2) Luther
(3232560085755078826, 2, 3) King
(3232560085755078826, 3, 4) Jr.
(3232560085755078826, 6, 7) Michael
(3232560085755078826, 7, 8) King
(3232560085755078826, 8, 9) Jr.
(3232560085755078826, 10, 11) January
(3232560085755078826, 14, 15) â€
(3232560085755078826, 16, 17) April
```

6.4.1. Improving it with Multi-Word Tokens

```
matcher = Matcher(nlp.vocab)
pattern = [{"POS": "PROPN", "OP": "+"}]
matcher.add("PROPER_NOUNS", [pattern])
doc = nlp(text)
matches = matcher(doc)
print (len(matches))
for match in matches[:10]:
    print (match, doc[match[1]:match[2]])
```

```
175
(3232560085755078826, 0, 1) Martin
(3232560085755078826, 0, 2) Martin Luther
(3232560085755078826, 1, 2) Luther
(3232560085755078826, 0, 3) Martin Luther King
(3232560085755078826, 1, 3) Luther King
(3232560085755078826, 2, 3) King
(3232560085755078826, 0, 4) Martin Luther King Jr.
(3232560085755078826, 1, 4) Luther King Jr.
(3232560085755078826, 2, 4) King Jr.
(3232560085755078826, 3, 4) Jr.
```

6.4.2. Greedy Keyword Argument

```
matcher = Matcher(nlp.vocab)
pattern = [{"POS": "PROPN", "OP": "+"}]
matcher.add("PROPER_NOUNS", [pattern], greedy='LONGEST')
doc = nlp(text)
matches = matcher(doc)
print (len(matches))
for match in matches[:10]:
    print (match, doc[match[1]:match[2]])
```

```
61
(3232560085755078826, 84, 89) Martin Luther King Sr.
(3232560085755078826, 470, 475) Martin Luther King Jr. Day
(3232560085755078826, 537, 542) Martin Luther King Jr. Memorial
(3232560085755078826, 0, 4) Martin Luther King Jr.
(3232560085755078826, 129, 133) Southern Christian Leadership Conference
(3232560085755078826, 248, 252) Director J. Edgar Hoover
(3232560085755078826, 6, 9) Michael King Jr.
(3232560085755078826, 326, 329) Nobel Peace Prize
(3232560085755078826, 423, 426) James Earl Ray
(3232560085755078826, 464, 467) Congressional Gold Medal
```

6.4.3. Sorting it to Apperance

```
matcher = Matcher(nlp.vocab)
pattern = [{"POS": "PROPN", "OP": "+"}]
matcher.add("PROPER_NOUNS", [pattern], greedy='LONGEST')
doc = nlp(text)
matches = matcher(doc)
matches.sort(key = lambda x: x[1])
print (len(matches))
for match in matches[:10]:
    print (match, doc[match[1]:match[2]])
```

```
61
(3232560085755078826, 0, 4) Martin Luther King Jr.
(3232560085755078826, 6, 9) Michael King Jr.
(3232560085755078826, 10, 11) January
(3232560085755078826, 14, 15) â€
(3232560085755078826, 16, 17) April
(3232560085755078826, 24, 25) Baptist
(3232560085755078826, 50, 51) King
(3232560085755078826, 70, 72) Mahatma Gandhi
(3232560085755078826, 84, 89) Martin Luther King Sr.
(3232560085755078826, 90, 91) King
```

6.4.4. Adding in Sequences

```
matcher = Matcher(nlp.vocab)
pattern = [{"POS": "PROPN", "OP": "+"}, {"POS": "VERB"}]
matcher.add("PROPER_NOUNS", [pattern], greedy='LONGEST')
doc = nlp(text)
matches = matcher(doc)
matches.sort(key = lambda x: x[1])
print (len(matches))
for match in matches[:10]:
    print (match, doc[match[1]:match[2]])
```

```
7
(3232560085755078826, 50, 52) King advanced
(3232560085755078826, 90, 92) King participated
(3232560085755078826, 114, 116) King led
(3232560085755078826, 168, 170) King helped
(3232560085755078826, 248, 253) Director J. Edgar Hoover considered
(3232560085755078826, 323, 325) King won
(3232560085755078826, 486, 489) United States beginning
```

6.5. Finding Quotes and Speakers

```
import json
with open ("data/alice.json", "r") as f:
    data = json.load(f)
```

```
text = data[0][2][0]
print (text)
```

```
Alice was beginning to get very tired of sitting by her sister on the bank, and of
having nothing to do: once or twice she had peeped into the book her sister was
reading, but it had no pictures or conversations in it, `and what is the use of a
book,' thought Alice `without pictures or conversation?'
```

```
text = data[0][2][0].replace( "`", "'")
print (text)
```

```
Alice was beginning to get very tired of sitting by her sister on the bank, and of
having nothing to do: once or twice she had peeped into the book her sister was
reading, but it had no pictures or conversations in it, 'and what is the use of a
book,' thought Alice 'without pictures or conversation?'
```

```
matcher = Matcher(nlp.vocab)
pattern = [{'ORTH': "'"}, {'IS_ALPHA': True, "OP": "+"}, {'IS_PUNCT': True, "OP":
"*"}, {'ORTH': "'"}]
matcher.add("PROPER_NOUNS", [pattern], greedy='LONGEST')
doc = nlp(text)
matches = matcher(doc)
matches.sort(key = lambda x: x[1])
print (len(matches))
for match in matches[:10]:
    print (match, doc[match[1]:match[2]])
```

```
2
(3232560085755078826, 47, 58) 'and what is the use of a book,'
(3232560085755078826, 60, 67) 'without pictures or conversation?'
```

6.5.1. Find Speaker

```
speak_lemmas = ["think", "say"]
text = data[0][2][0].replace( "`", "'")
matcher = Matcher(nlp.vocab)
pattern1 = [{'ORTH': "'"}, {'IS_ALPHA': True, "OP": "+"}, {'IS_PUNCT': True, "OP":
"*"}, {'ORTH': "'"}, {"POS": "VERB", "LEMMA": {"IN": speak_lemmas}}, {"POS": "PROPN",
"OP": "+"}, {'ORTH': "'"}, {'IS_ALPHA': True, "OP": "+"}, {'IS_PUNCT': True, "OP":
"*"}, {'ORTH': "'"}]
matcher.add("PROPER_NOUNS", [pattern1], greedy='LONGEST')
doc = nlp(text)
matches = matcher(doc)
matches.sort(key = lambda x: x[1])
print (len(matches))
for match in matches[:10]:
    print (match, doc[match[1]:match[2]])
```

```
1
(3232560085755078826, 47, 67) 'and what is the use of a book,' thought Alice 'without
pictures or conversation?'
```

6.5.2. Problem with this Approach

```
for text in data[0][2]:
    text = text.replace("`", "'")
    doc = nlp(text)
    matches = matcher(doc)
    matches.sort(key = lambda x: x[1])
    print (len(matches))
    for match in matches[:10]:
        print (match, doc[match[1]:match[2]])
```

```
1
(3232560085755078826, 47, 67) 'and what is the use of a book,' thought Alice 'without
pictures or conversation?'
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

6.5.3. Adding More Patterns

```
speak_lemmas = ["think", "say"]
text = data[0][2][0].replace( "`", "'")
matcher = Matcher(nlp.vocab)
pattern1 = [{'ORTH': "'"}, {'IS_ALPHA': True, "OP": "+"}, {'IS_PUNCT': True, "OP":
"*"}, {'ORTH': "'"}, {"POS": "VERB", "LEMMA": {"IN": speak_lemmas}}, {"POS": "PROPN",
"OP": "+"}, {'ORTH': "'"}, {'IS_ALPHA': True, "OP": "+"}, {'IS_PUNCT': True, "OP":
"*"}, {'ORTH': "'"}]
pattern2 = [{'ORTH': "'"}, {'IS_ALPHA': True, "OP": "+"}, {'IS_PUNCT': True, "OP":
"*"}, {'ORTH': "'"}, {"POS": "VERB", "LEMMA": {"IN": speak_lemmas}}, {"POS": "PROPN",
"OP": "+"}]
pattern3 = [{"POS": "PROPN", "OP": "+"},{"POS": "VERB", "LEMMA": {"IN": speak_lemmas}},
 {'ORTH': "'"}, {'IS_ALPHA': True, "OP": "+"}, {'IS_PUNCT': True, "OP": "*"}, {'ORTH':
"'"}]
matcher.add("PROPER_NOUNS", [pattern1, pattern2, pattern3], greedy='LONGEST')
for text in data[0][2]:
    text = text.replace("`", "'")
    doc = nlp(text)
    matches = matcher(doc)
    matches.sort(key = lambda x: x[1])
    print (len(matches))
    for match in matches[:10]:
        print (match, doc[match[1]:match[2]])
```

```
1
(3232560085755078826, 47, 67) 'and what is the use of a book,' thought Alice 'without
pictures or conversation?'
0
0
0
0
0
1
(3232560085755078826, 0, 6) 'Well!' thought Alice
0
0
0
0
0
0
0
1
(3232560085755078826, 57, 68) 'which certainly was not here before,' said Alice
0
0
```