gging-and-named-entity-recognition

January 27, 2024

[1]:
```
import spacy
```

[2]:
```
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords

corpus = 'India, officially the Republic of India (Hindi: Bhārat␣
 ↪Gaṇarājya),[25] is a country in South Asia. It is the seventh-largest␣
 ↪country by area, the second-most populous country, and the most populous␣
 ↪democracy in the world. Bounded by the Indian Ocean on the south, the␣
 ↪Arabian Sea on the southwest, and the Bay of Bengal on the southeast, it␣
 ↪shares land borders with Pakistan to the west;[f] China, Nepal, and Bhutan␣
 ↪to the north; and Bangladesh and Myanmar to the east. In the Indian Ocean,␣
 ↪India is in the vicinity of Sri Lanka and the Maldives; its Andaman and␣
 ↪Nicobar Islands share a maritime border with Thailand, Myanmar, and␣
 ↪Indonesia.'
```

[3]:
```
corpus
```

[3]:
```
'India, officially the Republic of India (Hindi: Bhārat Gaṇarājya),[25] is a
country in South Asia. It is the seventh-largest country by area, the second-
most populous country, and the most populous democracy in the world. Bounded by
the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of
Bengal on the southeast, it shares land borders with Pakistan to the west;[f]
China, Nepal, and Bhutan to the north; and Bangladesh and Myanmar to the east.
In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; its
Andaman and Nicobar Islands share a maritime border with Thailand, Myanmar, and
Indonesia.'
```

[4]:
```
nlp = spacy.load('en_core_web_lg')
```

0.1 Splitting The Tokens

[5]:
```
nlp('GFG is looking for data science')
```

[5]:
```
GFG is looking for data science
```

```
[6]: s = "GPT is one of the first of it's kind."

     d = nlp(s)
```

```
[7]: print(d[0])
```

```
GPT
```

```
[8]: print(d)
```

```
GPT is one of the first of it's kind.
```

```
[9]: d[0].text
```

```
[9]: 'GPT'
```

0.2 Finding Part Of Speech

```
[10]: d[0].pos_
```

```
[10]: 'PROPN'
```

```
[11]: d[4] , d[4].pos_
```

```
[11]: (the, 'DET')
```

```
[12]: d[6] , d[6].pos_
```

```
[12]: (of, 'ADP')
```

```
[13]: for i in range(0,len(d)):
          print(d[i], " : ", d[i].pos_)
```

```
GPT  :  PROPN
is  :  AUX
one  :  NUM
of  :  ADP
the  :  DET
first  :  ADJ
of  :  ADP
it  :  PRON
's  :  AUX
kind  :  ADJ
.  :  PUNCT
```

0.3 Finding Grained Part Of Speech

[14]:
```
for i in range(0,len(d)):
    print(d[i], " : ", d[i].tag_)
```

```
GPT  :  NNP
is  :  VBZ
one  :  CD
of  :  IN
the  :  DT
first  :  JJ
of  :  IN
it  :  PRP
's  :  VBZ
kind  :  JJ
.  :  .
```

[15]:
```
for token in d:
    print(token)
```

```
GPT
is
one
of
the
first
of
it
's
kind
.
```

0.4 spacy.explain

[16]:
```
for token in d:
    print(f'{token.text:{15}}{token.pos_:{15}}{token.tag_:{15}}{spacy.
↪explain(token.tag_)}')
```

```
GPT            PROPN          NNP            noun, proper singular
is             AUX            VBZ            verb, 3rd person singular present
one            NUM            CD             cardinal number
of             ADP            IN             conjunction, subordinating or
preposition
the            DET            DT             determiner
first          ADJ            JJ             adjective (English), other noun-
modifier (Chinese)
of             ADP            IN             conjunction, subordinating or
preposition
```

```
it              PRON            PRP             pronoun, personal
's              AUX             VBZ             verb, 3rd person singular present
kind            ADJ             JJ              adjective (English), other noun-
modifier (Chinese)
.               PUNCT           .               punctuation mark, sentence closer
```

[17]:
```
spacy.explain(d[0].tag_)
```

[17]:
```
'noun, proper singular'
```

0.5 Visualisation Of Part Of Speech

Displacy is tool for visualisation of part of speech.

displacy.render(document) : This helps in visualising the POS (Part Of Speech) of the document.

[18]:
```
from spacy import displacy
```

[19]:
```
displacy.render(d, jupyter = True)
```

```
<IPython.core.display.HTML object>
```

[20]:
```
displacy.render(d)
```

```
<IPython.core.display.HTML object>
```

[21]:
```
displacy.render(d, jupyter = True, options = {'distance' : 100, 'color' :␣
 ↪'white', 'bg' : 'red', 'font' : 'times'})
```

```
<IPython.core.display.HTML object>
```

0.6 Named Identity Recognition

[22]:
```
doc_2 = nlp('Disney Princess, also called the Princess Line, is a media␣
 ↪franchise and toy line owned by the Walt Disney Company. Created by Disney␣
 ↪Consumer Products chairman Andy Mooney, the franchise features a lineup of␣
 ↪female protagonists who have appeared in various Disney franchises.')
doc_2
```

[22]:
```
Disney Princess, also called the Princess Line, is a media franchise and toy
line owned by the Walt Disney Company. Created by Disney Consumer Products
chairman Andy Mooney, the franchise features a lineup of female protagonists who
have appeared in various Disney franchises.
```

[23]:
```
doc_2.ents
```

[23]:
```
(Disney Princess,
 the Princess Line,
 the Walt Disney Company,
 Disney Consumer Products,
 Andy Mooney,
 Disney)
```

[24]:
```
def show_entities(doc):
    if doc_2.ents:
        for ent in doc.ents:
            print(f'{ent.text:{30}}{ent.label_:{30}}{spacy.explain(ent.
 ↪label_)}')
    else:
        print('No entities Found')
```

[25]:
```
show_entities(doc_2)
```

```
Disney Princess              ORG                           Companies, agencies,
institutions, etc.
the Princess Line            PRODUCT                       Objects, vehicles,
foods, etc. (not services)
the Walt Disney Company      ORG                           Companies, agencies,
institutions, etc.
Disney Consumer Products     ORG                           Companies, agencies,
institutions, etc.
Andy Mooney                  PERSON                        People, including
fictional
Disney                       ORG                           Companies, agencies,
institutions, etc.
```

[26]:
```
show_entities(nlp("I'm not feeling well today."))
```

```
today                        DATE                          Absolute or relative
dates or periods
```

[27]:
```
show_entities(nlp("The earth revolves around the sun in 24 hours."))
show_entities(nlp("The radius of the earth is 6371 kilometres."))
```

```
24 hours                     TIME                          Times smaller than a
day
6371 kilometres              QUANTITY                      Measurements, as of
weight or distance
```

[28]:
```
show_entities(nlp("The earth revolves around the sun."))
```

0.7 Adding New Entity

```
[29]: show_entities(nlp("Tesla is one of the biggest giant in the field of electric␣
        ↪vehicles."))
```

```
Tesla                         ORG                           Companies, agencies,
institutions, etc.
```

```
[30]: from spacy.tokens import Span as sp

      d1 = nlp("Earth revolves around the sun.")
      d2 = nlp("Tesla is one of the biggest giant in the field of electric vehicles.")

      ORG = d2.vocab.strings['ORG']
      new_entity1 = sp(d2, 0, 1, label = d1.vocab.strings['ORG'])
```

```
[31]: [new_entity1]
```

```
[31]: [Tesla]
```

0.8 Adding New Multiple Entities at a time

```
[32]: d2 = nlp("Playing Cricket and Football are good enough for health.")

      show_entities(d2)
```

```
[33]: from spacy.matcher import PhraseMatcher
      m = PhraseMatcher(nlp.vocab)
      phrase = ['Cricket','Football']
```

```
[34]: print([nlp(text) for text in phrase])
```

```
[Cricket, Football]
```

```
[35]: pattern = [nlp(text) for text in phrase]
      m.add('Sports', None, *pattern)
      m(d2)
```

```
[35]: [(9611670226552988807, 1, 2), (9611670226552988807, 3, 4)]
```

```
[36]: from spacy.tokens import Span as sp

      sport = d2.vocab.strings['Sports']

      found = m(d2)
```

```
[37]: for mtch in found:
          print(mtch[1], mtch[2])
```

```
1 2
3 4
```

```
[38]: new_ents = [sp(d2, mtch[1], mtch[2], label = sport) for mtch in found]
      print(new_ents)
```

```
[Cricket, Football]
```

```
[39]: d2.ents = list(d2.ents) + new_ents
```

```
[40]: print(d2.ents)
```

```
(Cricket, Football)
```

```
[41]: show_entities(d2)
```

```
Cricket                       Sports                        None
Football                      Sports                        None

C:\Users\Soubhik\anaconda3\Anaconda\lib\site-packages\spacy\glossary.py:20:
UserWarning: [W118] Term 'Sports' not found in glossary. It may however be
explained in documentation for the corpora used to train the language. Please
check `nlp.meta["sources"]` for any relevant links.
  warnings.warn(Warnings.W118.format(term=term))
```

0.9 Combining Two corpuses and Performing NER

```
[42]: d3 = nlp("Tesla is one of the biggest giant in the field of electric vehicles.␣
        ↪Playing Cricket and Football are good enough for health.")
      show_entities(d3)
```

```
Tesla                         ORG                           Companies, agencies,
institutions, etc.
```

```
[43]: m = PhraseMatcher(nlp.vocab)
      phrase = ['Cricket','Football']
      pattern = [nlp(text) for text in phrase]
      m.add('Sports', None, *pattern)
      sport = d3.vocab.strings['Sports']
      found = m(d3)
      new_ents = [sp(d3, mtch[1], mtch[2], label = sport) for mtch in found]
      d3.ents = list(d3.ents) + new_ents
```

```
[44]: show_entities(d3)
```

```
Tesla                         ORG                           Companies, agencies,
institutions, etc.
Cricket                       Sports                        None
Football                      Sports                        None
```

0.10 Finding Specific Tag Entities

[45]:
```
d4 = nlp("Google, Apple, GFG, Tesla, Grapes are good.")
show_entities(d4)
```

```
Google                        ORG                           Companies, agencies,
institutions, etc.
Apple                         ORG                           Companies, agencies,
institutions, etc.
GFG                           ORG                           Companies, agencies,
institutions, etc.
Tesla                         ORG                           Companies, agencies,
institutions, etc.
```

[46]:
```
d4 = nlp("Google, Apple, GFG, student")
show_entities(d4)
```

```
Google                        ORG                           Companies, agencies,
institutions, etc.
Apple                         ORG                           Companies, agencies,
institutions, etc.
GFG                           ORG                           Companies, agencies,
institutions, etc.
```

[47]:
```
[ent for ent in d4.ents]
```

[47]:
```
[Google, Apple, GFG]
```

[48]:
```
[ent for ent in d4.ents if ent.label_ == 'ORG']
```

[48]:
```
[Google, Apple, GFG]
```

[49]:
```
d5 = nlp("Google, Apple, GFG, 1 Million, Phone, 2 Million Dollars")
show_entities(d5)
```

```
Google                        ORG                           Companies, agencies,
institutions, etc.
Apple                         ORG                           Companies, agencies,
institutions, etc.
GFG                           ORG                           Companies, agencies,
institutions, etc.
1 Million                     CARDINAL                      Numerals that do not
fall under another type
```

```
2 Million Dollars              MONEY                          Monetary values,
including unit
```

[50]:
```python
print([ent for ent in d5.ents if ent.label_ == 'CARDINAL'])
print('\n')
print([ent for ent in d5.ents if ent.label_ == 'MONEY'])
```

```
[1 Million]


[2 Million Dollars]
```

0.11 REFERENCES OR APPENDIX

[51]:
```python
help(nlp)
```

```
Help on English in module spacy.lang.en object:

class English(spacy.language.Language)
 |  English(vocab: Union[spacy.vocab.Vocab, bool] = True, *, max_length: int =
1000000, meta: Dict[str, Any] = {}, create_tokenizer:
Optional[Callable[[ForwardRef('Language')], Callable[[str],
spacy.tokens.doc.Doc]]] = None, create_vectors:
Optional[Callable[[ForwardRef('Vocab')], spacy.vectors.BaseVectors]] = None,
batch_size: int = 1000, **kwargs) -> None
 |
 |  Method resolution order:
 |      English
 |      spacy.language.Language
 |      builtins.object
 |
 |  Data and other attributes defined here:
 |
 |  Defaults = <class 'spacy.lang.en.EnglishDefaults'>
 |
 |  default_config = {'paths': {'train': None, 'dev': None, 'vectors'…s'…
 |
 |  factories = {'attribute_ruler': <function make_attribute_rul…<functi…
 |
 |  lang = 'en'
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from spacy.language.Language:
 |
 |  __call__(self, text: Union[str, spacy.tokens.doc.Doc], *, disable:
Iterable[str] = [], component_cfg: Optional[Dict[str, Dict[str, Any]]] = None)
-> spacy.tokens.doc.Doc
 |      Apply the pipeline to some text. The text can span multiple sentences,
 |      and can contain arbitrary whitespace. Alignment into the original string
 |      is preserved.
 |
 |      text (Union[str, Doc]): If `str`, the text to be processed. If `Doc`,
 |          the doc will be passed directly to the pipeline, skipping
 |          `Language.make_doc`.
 |      disable (List[str]): Names of the pipeline components to disable.
 |      component_cfg (Dict[str, dict]): An optional dictionary with extra
 |          keyword arguments for specific components.
 |      RETURNS (Doc): A container for accessing the annotations.
 |
 |      DOCS: https://spacy.io/api/language#call
 |
 |  __init__(self, vocab: Union[spacy.vocab.Vocab, bool] = True, *, max_length:
int = 1000000, meta: Dict[str, Any] = {}, create_tokenizer:
Optional[Callable[[ForwardRef('Language')], Callable[[str],
spacy.tokens.doc.Doc]]] = None, create_vectors:
Optional[Callable[[ForwardRef('Vocab')], spacy.vectors.BaseVectors]] = None,
batch_size: int = 1000, **kwargs) -> None
 |      Initialise a Language object.
 |
 |      vocab (Vocab): A `Vocab` object. If `True`, a vocab is created.
 |      meta (dict): Custom meta data for the Language class. Is written to by
 |          models to add model meta data.
 |      max_length (int): Maximum number of characters in a single text. The
 |          current models may run out memory on extremely long texts, due to
 |          large internal allocations. You should segment these texts into
 |          meaningful units, e.g. paragraphs, subsections etc, before passing
 |          them to spaCy. Default maximum length is 1,000,000 charas (1mb). As
 |          a rule of thumb, if all pipeline components are enabled, spaCy's
 |          default models currently requires roughly 1GB of temporary memory
per
 |          100,000 characters in one text.
 |      create_tokenizer (Callable): Function that takes the nlp object and
 |          returns a tokenizer.
 |      batch_size (int): Default batch size for pipe and evaluate.
 |
 |      DOCS: https://spacy.io/api/language#init
 |
 |  add_pipe(self, factory_name: str, name: Optional[str] = None, *, before:
Union[str, int, NoneType] = None, after: Union[str, int, NoneType] = None,
first: Optional[bool] = None, last: Optional[bool] = None, source:
Optional[ForwardRef('Language')] = None, config: Dict[str, Any] = {},
raw_config: Optional[confection.Config] = None, validate: bool = True) ->
Callable[[spacy.tokens.doc.Doc], spacy.tokens.doc.Doc]
 |      Add a component to the processing pipeline. Valid components are
 |      callables that take a `Doc` object, modify it and return it. Only one
 |      of before/after/first/last can be set. Default behaviour is "last".
 |
 |      factory_name (str): Name of the component factory.
 |      name (str): Name of pipeline component. Overwrites existing
 |          component.name attribute if available. If no name is set and
 |          the component exposes no name attribute, component.__name__ is
 |          used. An error is raised if a name already exists in the pipeline.
 |      before (Union[str, int]): Name or index of the component to insert new
 |          component directly before.
 |      after (Union[str, int]): Name or index of the component to insert new
 |          component directly after.
 |      first (bool): If True, insert component first in the pipeline.
 |      last (bool): If True, insert component last in the pipeline.
 |      source (Language): Optional loaded nlp object to copy the pipeline
 |          component from.
 |      config (Dict[str, Any]): Config parameters to use for this component.
 |          Will be merged with default config, if available.
 |      raw_config (Optional[Config]): Internals: the non-interpolated config.
 |      validate (bool): Whether to validate the component config against the
 |          arguments and types expected by the factory.
 |      RETURNS (Callable[[Doc], Doc]): The pipeline component.
 |
 |      DOCS: https://spacy.io/api/language#add_pipe
 |
 |  analyze_pipes(self, *, keys: List[str] = ['assigns', 'requires', 'scores',
'retokenizes'], pretty: bool = False) -> Optional[Dict[str, Any]]
 |      Analyze the current pipeline components, print a summary of what
 |      they assign or require and check that all requirements are met.
 |
 |      keys (List[str]): The meta values to display in the table. Corresponds
 |          to values in FactoryMeta, defined by @Language.factory decorator.
 |      pretty (bool): Pretty-print the results.
 |      RETURNS (dict): The data.
 |
 |  begin_training(self, get_examples: Optional[Callable[[],
Iterable[spacy.training.example.Example]]] = None, *, sgd:
Optional[thinc.optimizers.Optimizer] = None) -> thinc.optimizers.Optimizer
 |
 |  create_optimizer(self)
 |      Create an optimizer, usually using the [training.optimizer] config.
 |
 |  create_pipe(self, factory_name: str, name: Optional[str] = None, *, config:
Dict[str, Any] = {}, raw_config: Optional[confection.Config] = None, validate:
bool = True) -> Callable[[spacy.tokens.doc.Doc], spacy.tokens.doc.Doc]
 |      Create a pipeline component. Mostly used internally. To create and
 |      add a component to the pipeline, you can use nlp.add_pipe.
 |
 |      factory_name (str): Name of component factory.
 |      name (Optional[str]): Optional name to assign to component instance.
 |          Defaults to factory name if not set.
 |      config (Dict[str, Any]): Config parameters to use for this component.
 |          Will be merged with default config, if available.
 |      raw_config (Optional[Config]): Internals: the non-interpolated config.
 |      validate (bool): Whether to validate the component config against the
 |          arguments and types expected by the factory.
 |      RETURNS (Callable[[Doc], Doc]): The pipeline component.
 |
 |      DOCS: https://spacy.io/api/language#create_pipe
 |
 |  create_pipe_from_source(self, source_name: str, source: 'Language', *, name:
str) -> Tuple[Callable[[spacy.tokens.doc.Doc], spacy.tokens.doc.Doc], str]
 |      Create a pipeline component by copying it from an existing model.
 |
 |      source_name (str): Name of the component in the source pipeline.
 |      source (Language): The source nlp object to copy from.
 |      name (str): Optional alternative name to use in current pipeline.
 |      RETURNS (Tuple[Callable[[Doc], Doc], str]): The component and its
factory name.
 |
 |  disable_pipe(self, name: str) -> None
 |      Disable a pipeline component. The component will still exist on
 |      the nlp object, but it won't be run as part of the pipeline. Does
 |      nothing if the component is already disabled.
 |
 |      name (str): The name of the component to disable.
 |
 |  disable_pipes(self, *names) -> 'DisabledPipes'
 |      Disable one or more pipeline components. If used as a context
 |      manager, the pipeline will be restored to the initial state at the end
 |      of the block. Otherwise, a DisabledPipes object is returned, that has
 |      a `.restore()` method you can use to undo your changes.
 |
 |      This method has been deprecated since 3.0
 |
 |  enable_pipe(self, name: str) -> None
 |      Enable a previously disabled pipeline component so it's run as part
 |      of the pipeline. Does nothing if the component is already enabled.
 |
 |      name (str): The name of the component to enable.
 |
 |  evaluate(self, examples: Iterable[spacy.training.example.Example], *,
batch_size: Optional[int] = None, scorer: Optional[spacy.scorer.Scorer] = None,
component_cfg: Optional[Dict[str, Dict[str, Any]]] = None, scorer_cfg:
Optional[Dict[str, Any]] = None, per_component: bool = False) -> Dict[str, Any]
 |      Evaluate a model's pipeline components.
 |
 |      examples (Iterable[Example]): `Example` objects.
 |      batch_size (Optional[int]): Batch size to use.
 |      scorer (Optional[Scorer]): Scorer to use. If not passed in, a new one
 |          will be created.
 |      component_cfg (dict): An optional dictionary with extra keyword
 |          arguments for specific components.
 |      scorer_cfg (dict): An optional dictionary with extra keyword arguments
 |          for the scorer.
 |      per_component (bool): Whether to return the scores keyed by component
 |          name. Defaults to False.
 |
 |      RETURNS (Scorer): The scorer containing the evaluation results.
 |
 |      DOCS: https://spacy.io/api/language#evaluate
 |
 |  from_bytes(self, bytes_data: bytes, *, exclude: Iterable[str] = []) ->
'Language'
 |      Load state from a binary string.
 |
 |      bytes_data (bytes): The data to load from.
 |      exclude (Iterable[str]): Names of components or serialization fields to
exclude.
 |      RETURNS (Language): The `Language` object.
 |
 |      DOCS: https://spacy.io/api/language#from_bytes
 |
 |  from_disk(self, path: Union[str, pathlib.Path], *, exclude: Iterable[str] =
[], overrides: Dict[str, Any] = {}) -> 'Language'
 |      Loads state from a directory. Modifies the object in place and
 |      returns it. If the saved `Language` object contains a model, the
 |      model will be loaded.
 |
 |      path (str / Path): A path to a directory.
 |      exclude (Iterable[str]): Names of components or serialization fields to
exclude.
 |      RETURNS (Language): The modified `Language` object.
 |
 |      DOCS: https://spacy.io/api/language#from_disk
 |
 |  get_pipe(self, name: str) -> Callable[[spacy.tokens.doc.Doc],
spacy.tokens.doc.Doc]
 |      Get a pipeline component for a given component name.
 |
 |      name (str): Name of pipeline component to get.
 |      RETURNS (callable): The pipeline component.
 |
 |      DOCS: https://spacy.io/api/language#get_pipe
 |
 |  get_pipe_config(self, name: str) -> confection.Config
 |      Get the config used to create a pipeline component.
 |
 |      name (str): The component name.
 |      RETURNS (Config): The config used to create the pipeline component.
 |
 |  get_pipe_meta(self, name: str) -> 'FactoryMeta'
 |      Get the meta information for a given component name.
 |
 |      name (str): The component name.
 |      RETURNS (FactoryMeta): The meta for the given component name.
 |
 |  has_pipe(self, name: str) -> bool
 |      Check if a component name is present in the pipeline. Equivalent to
 |      `name in nlp.pipe_names`.
 |
 |      name (str): Name of the component.
 |      RETURNS (bool): Whether a component of the name exists in the pipeline.
 |
 |      DOCS: https://spacy.io/api/language#has_pipe
 |
 |  initialize(self, get_examples: Optional[Callable[[],
Iterable[spacy.training.example.Example]]] = None, *, sgd:
Optional[thinc.optimizers.Optimizer] = None) -> thinc.optimizers.Optimizer
 |      Initialize the pipe for training, using data examples if available.
 |
 |      get_examples (Callable[[], Iterable[Example]]): Optional function that
 |          returns gold-standard Example objects.
 |      sgd (Optional[Optimizer]): An optimizer to use for updates. If not
 |          provided, will be created using the .create_optimizer() method.
 |      RETURNS (thinc.api.Optimizer): The optimizer.
 |
 |      DOCS: https://spacy.io/api/language#initialize
 |
 |  make_doc(self, text: str) -> spacy.tokens.doc.Doc
 |      Turn a text into a Doc object.
 |
 |      text (str): The text to process.
 |      RETURNS (Doc): The processed doc.
 |
 |  pipe(self, texts: Union[Iterable[Union[str, spacy.tokens.doc.Doc]],
Iterable[Tuple[Union[str, spacy.tokens.doc.Doc], ~_AnyContext]]], *, as_tuples:
bool = False, batch_size: Optional[int] = None, disable: Iterable[str] = [],
component_cfg: Optional[Dict[str, Dict[str, Any]]] = None, n_process: int = 1)
-> Union[Iterator[spacy.tokens.doc.Doc], Iterator[Tuple[spacy.tokens.doc.Doc,
~_AnyContext]]]
 |      Process texts as a stream, and yield `Doc` objects in order.
 |
 |      texts (Iterable[Union[str, Doc]]): A sequence of texts or docs to
 |          process.
 |      as_tuples (bool): If set to True, inputs should be a sequence of
 |          (text, context) tuples. Output will then be a sequence of
 |          (doc, context) tuples. Defaults to False.
 |      batch_size (Optional[int]): The number of texts to buffer.
 |      disable (List[str]): Names of the pipeline components to disable.
 |      component_cfg (Dict[str, Dict]): An optional dictionary with extra
keyword
 |          arguments for specific components.
 |      n_process (int): Number of processors to process texts. If -1, set
`multiprocessing.cpu_count()`.
 |      YIELDS (Doc): Documents in the order of the original text.
 |
 |      DOCS: https://spacy.io/api/language#pipe
 |
 |  rehearse(self, examples: Iterable[spacy.training.example.Example], *, sgd:
Optional[thinc.optimizers.Optimizer] = None, losses: Optional[Dict[str, float]]
= None, component_cfg: Optional[Dict[str, Dict[str, Any]]] = None, exclude:
Iterable[str] = []) -> Dict[str, float]
 |      Make a "rehearsal" update to the models in the pipeline, to prevent
 |      forgetting. Rehearsal updates run an initial copy of the model over some
 |      data, and update the model so its current predictions are more like the
 |      initial ones. This is useful for keeping a pretrained model on-track,
 |      even if you're updating it with a smaller set of examples.
 |
 |      examples (Iterable[Example]): A batch of `Example` objects.
 |      sgd (Optional[Optimizer]): An optimizer.
 |      component_cfg (Dict[str, Dict]): Config parameters for specific pipeline
 |          components, keyed by component name.
 |      exclude (Iterable[str]): Names of components that shouldn't be updated.
 |      RETURNS (dict): Results from the update.
 |
 |      EXAMPLE:
 |          >>> raw_text_batches = minibatch(raw_texts)
 |          >>> for labelled_batch in minibatch(examples):
 |          >>>     nlp.update(labelled_batch)
 |          >>>     raw_batch = [Example.from_dict(nlp.make_doc(text), {}) for
text in next(raw_text_batches)]
 |          >>>     nlp.rehearse(raw_batch)
 |
 |      DOCS: https://spacy.io/api/language#rehearse
 |
 |  remove_pipe(self, name: str) -> Tuple[str, Callable[[spacy.tokens.doc.Doc],
spacy.tokens.doc.Doc]]
 |      Remove a component from the pipeline.
 |
 |      name (str): Name of the component to remove.
 |      RETURNS (Tuple[str, Callable[[Doc], Doc]]): A `(name, component)` tuple
```

```
of the removed component.
 |
 |      DOCS: https://spacy.io/api/language#remove_pipe
 |
 |  rename_pipe(self, old_name: str, new_name: str) -> None
 |      Rename a pipeline component.
 |
 |      old_name (str): Name of the component to rename.
 |      new_name (str): New name of the component.
 |
 |      DOCS: https://spacy.io/api/language#rename_pipe
 |
 |  replace_listeners(self, tok2vec_name: str, pipe_name: str, listeners:
Iterable[str]) -> None
 |      Find listener layers (connecting to a token-to-vector embedding
 |      component) of a given pipeline component model and replace
 |      them with a standalone copy of the token-to-vector layer. This can be
 |      useful when training a pipeline with components sourced from an existing
 |      pipeline: if multiple components (e.g. tagger, parser, NER) listen to
 |      the same tok2vec component, but some of them are frozen and not updated,
 |      their performance may degrade significantly as the tok2vec component is
 |      updated with new data. To prevent this, listeners can be replaced with
 |      a standalone tok2vec layer that is owned by the component and doesn't
 |      change if the component isn't updated.
 |
 |      tok2vec_name (str): Name of the token-to-vector component, typically
 |          "tok2vec" or "transformer".
 |      pipe_name (str): Name of pipeline component to replace listeners for.
 |      listeners (Iterable[str]): The paths to the listeners, relative to the
 |          component config, e.g. ["model.tok2vec"]. Typically, implementations
 |          will only connect to one tok2vec component, [model.tok2vec], but in
 |          theory, custom models can use multiple listeners. The value here can
 |          either be an empty list to not replace any listeners, or a complete
 |          (!) list of the paths to all listener layers used by the model.
 |
 |      DOCS: https://spacy.io/api/language#replace_listeners
 |
 |  replace_pipe(self, name: str, factory_name: str, *, config: Dict[str, Any] =
{}, validate: bool = True) -> Callable[[spacy.tokens.doc.Doc],
spacy.tokens.doc.Doc]
 |      Replace a component in the pipeline.
 |
 |      name (str): Name of the component to replace.
 |      factory_name (str): Factory name of replacement component.
 |      config (Optional[Dict[str, Any]]): Config parameters to use for this
 |          component. Will be merged with default config, if available.
 |      validate (bool): Whether to validate the component config against the
 |          arguments and types expected by the factory.
 |      RETURNS (Callable[[Doc], Doc]): The new pipeline component.
 |
 |      DOCS: https://spacy.io/api/language#replace_pipe
 |
 |  resume_training(self, *, sgd: Optional[thinc.optimizers.Optimizer] = None)
-> thinc.optimizers.Optimizer
 |      Continue training a pretrained model.
 |
 |      Create and return an optimizer, and initialize "rehearsal" for any
pipeline
 |      component that has a .rehearse() method. Rehearsal is used to prevent
 |      models from "forgetting" their initialized "knowledge". To perform
 |      rehearsal, collect samples of text you want the models to retain
performance
 |      on, and call nlp.rehearse() with a batch of Example objects.
 |
 |      RETURNS (Optimizer): The optimizer.
 |
 |      DOCS: https://spacy.io/api/language#resume_training
 |
 |  select_pipes(self, *, disable: Union[str, Iterable[str], NoneType] = None,
enable: Union[str, Iterable[str], NoneType] = None) -> 'DisabledPipes'
 |      Disable one or more pipeline components. If used as a context
 |      manager, the pipeline will be restored to the initial state at the end
 |      of the block. Otherwise, a DisabledPipes object is returned, that has
 |      a `.restore()` method you can use to undo your changes.
 |
 |      disable (str or iterable): The name(s) of the pipes to disable
 |      enable (str or iterable): The name(s) of the pipes to enable - all
others will be disabled
 |
 |      DOCS: https://spacy.io/api/language#select_pipes
 |
 |  set_error_handler(self, error_handler: Callable[[str,
Callable[[spacy.tokens.doc.Doc], spacy.tokens.doc.Doc],
List[spacy.tokens.doc.Doc], Exception], NoReturn])
 |      Set an error handler object for all the components in the pipeline
 |      that implement a set_error_handler function.
 |
 |      error_handler (Callable[[str, Callable[[Doc], Doc], List[Doc],
Exception], NoReturn]):
 |          Function that deals with a failing batch of documents. This callable
 |          function should take in the component's name, the component itself,
 |          the offending batch of documents, and the exception that was thrown.
 |      DOCS: https://spacy.io/api/language#set_error_handler
 |
 |  to_bytes(self, *, exclude: Iterable[str] = []) -> bytes
 |      Serialize the current state to a binary string.
 |
 |      exclude (Iterable[str]): Names of components or serialization fields to
exclude.
 |      RETURNS (bytes): The serialized form of the `Language` object.
 |
 |      DOCS: https://spacy.io/api/language#to_bytes
 |
 |  to_disk(self, path: Union[str, pathlib.Path], *, exclude: Iterable[str] =
[]) -> None
 |      Save the current state to a directory.  If a model is loaded, this
 |      will include the model.
 |
 |      path (str / Path): Path to a directory, which will be created if
 |          it doesn't exist.
 |      exclude (Iterable[str]): Names of components or serialization fields to
exclude.
 |
 |      DOCS: https://spacy.io/api/language#to_disk
 |
 |  update(self, examples: Iterable[spacy.training.example.Example], _:
Optional[Any] = None, *, drop: float = 0.0, sgd:
Optional[thinc.optimizers.Optimizer] = None, losses: Optional[Dict[str, float]]
= None, component_cfg: Optional[Dict[str, Dict[str, Any]]] = None, exclude:
Iterable[str] = [], annotates: Iterable[str] = [])
 |      Update the models in the pipeline.
 |
 |      examples (Iterable[Example]): A batch of examples
 |      _: Should not be set - serves to catch backwards-incompatible scripts.
 |      drop (float): The dropout rate.
 |      sgd (Optimizer): An optimizer.
 |      losses (Dict[str, float]): Dictionary to update with the loss, keyed by
 |          component.
 |      component_cfg (Dict[str, Dict]): Config parameters for specific pipeline
 |          components, keyed by component name.
 |      exclude (Iterable[str]): Names of components that shouldn't be updated.
 |      annotates (Iterable[str]): Names of components that should set
 |          annotations on the predicted examples after updating.
 |      RETURNS (Dict[str, float]): The updated losses dictionary
 |
 |      DOCS: https://spacy.io/api/language#update
 |
 |  use_params(self, params: Optional[dict])
 |      Replace weights of models in the pipeline with those provided in the
 |      params dictionary. Can be used as a contextmanager, in which case,
 |      models go back to their original weights after the block.
 |
 |      params (dict): A dictionary of parameters keyed by model ID.
 |
 |      EXAMPLE:
 |          >>> with nlp.use_params(optimizer.averages):
 |          >>>     nlp.to_disk("/tmp/checkpoint")
 |
 |      DOCS: https://spacy.io/api/language#use_params
 |
 |  ----------------------------------------------------------------------
 |  Class methods inherited from spacy.language.Language:
 |
 |  __init_subclass__(**kwargs) from builtins.type
 |      This method is called when a class is subclassed.
 |
 |      The default implementation does nothing. It may be
 |      overridden to extend subclasses.
 |
 |  component(name: str, *, assigns: Iterable[str] = [], requires: Iterable[str]
= [], retokenizes: bool = False, func: Optional[Callable[[spacy.tokens.doc.Doc],
spacy.tokens.doc.Doc]] = None) -> Callable[…, Any] from builtins.type
 |      Register a new pipeline component. Can be used for stateless function
 |      components that don't require a separate factory. Can be used as a
 |      decorator on a function or classmethod, or called as a function with the
 |      factory provided as the func keyword argument. To create a component and
 |      add it to the pipeline, you can use nlp.add_pipe(name).
 |
 |      name (str): The name of the component factory.
 |      assigns (Iterable[str]): Doc/Token attributes assigned by this
component,
 |          e.g. "token.ent_id". Used for pipeline analysis.
 |      requires (Iterable[str]): Doc/Token attributes required by this
component,
 |          e.g. "token.ent_id". Used for pipeline analysis.
 |      retokenizes (bool): Whether the component changes the tokenization.
 |          Used for pipeline analysis.
 |      func (Optional[Callable[[Doc], Doc]): Factory function if not used as a
decorator.
 |
 |      DOCS: https://spacy.io/api/language#component
 |
 |  factory(name: str, *, default_config: Dict[str, Any] = {}, assigns:
Iterable[str] = [], requires: Iterable[str] = [], retokenizes: bool = False,
default_score_weights: Dict[str, Optional[float]] = {}, func: Optional[Callable]
= None) -> Callable from builtins.type
 |      Register a new pipeline component factory. Can be used as a decorator
 |      on a function or classmethod, or called as a function with the factory
 |      provided as the func keyword argument. To create a component and add
 |      it to the pipeline, you can use nlp.add_pipe(name).
 |
 |      name (str): The name of the component factory.
 |      default_config (Dict[str, Any]): Default configuration, describing the
 |          default values of the factory arguments.
 |      assigns (Iterable[str]): Doc/Token attributes assigned by this
component,
 |          e.g. "token.ent_id". Used for pipeline analysis.
 |      requires (Iterable[str]): Doc/Token attributes required by this
component,
 |          e.g. "token.ent_id". Used for pipeline analysis.
 |      retokenizes (bool): Whether the component changes the tokenization.
 |          Used for pipeline analysis.
 |      default_score_weights (Dict[str, Optional[float]]): The scores to report
during
 |          training, and their default weight towards the final score used to
 |          select the best model. Weights should sum to 1.0 per component and
 |          will be combined and normalized for the whole pipeline. If None,
 |          the score won't be shown in the logs or be weighted.
 |      func (Optional[Callable]): Factory function if not used as a decorator.
 |
 |      DOCS: https://spacy.io/api/language#factory
 |
 |  from_config(config: Union[Dict[str, Any], confection.Config] = {}, *, vocab:
Union[spacy.vocab.Vocab, bool] = True, disable: Union[str, Iterable[str]] = [],
enable: Union[str, Iterable[str]] = [], exclude: Union[str, Iterable[str]] = [],
meta: Dict[str, Any] = {}, auto_fill: bool = True, validate: bool = True) ->
'Language' from builtins.type
 |      Create the nlp object from a loaded config. Will set up the tokenizer
 |      and language data, add pipeline components etc. If no config is
provided,
 |      the default config of the given language is used.
 |
 |      config (Dict[str, Any] / Config): The loaded config.
 |      vocab (Vocab): A Vocab object. If True, a vocab is created.
 |      disable (Union[str, Iterable[str]]): Name(s) of pipeline component(s) to
disable.
 |          Disabled pipes will be loaded but they won't be run unless you
 |          explicitly enable them by calling nlp.enable_pipe.
 |      enable (Union[str, Iterable[str]]): Name(s) of pipeline component(s) to
enable. All other
 |          pipes will be disabled (and can be enabled using `nlp.enable_pipe`).
 |      exclude (Union[str, Iterable[str]]): Name(s) of pipeline component(s) to
exclude.
 |          Excluded components won't be loaded.
 |      meta (Dict[str, Any]): Meta overrides for nlp.meta.
 |      auto_fill (bool): Automatically fill in missing values in config based
 |          on defaults and function argument annotations.
 |      validate (bool): Validate the component config and arguments against
 |          the types expected by the factory.
 |      RETURNS (Language): The initialized Language class.
 |
 |      DOCS: https://spacy.io/api/language#from_config
 |
 |  get_factory_meta(name: str) -> 'FactoryMeta' from builtins.type
 |      Get the meta information for a given factory name.
 |
 |      name (str): The component factory name.
 |      RETURNS (FactoryMeta): The meta for the given factory name.
 |
 |  get_factory_name(name: str) -> str from builtins.type
 |      Get the internal factory name based on the language subclass.
 |
 |      name (str): The factory name.
 |      RETURNS (str): The internal factory name.
 |
 |  has_factory(name: str) -> bool from builtins.type
 |      RETURNS (bool): Whether a factory of that name is registered.
 |
 |  set_factory_meta(name: str, value: 'FactoryMeta') -> None from builtins.type
 |      Set the meta information for a given factory name.
 |
 |      name (str): The component factory name.
 |      value (FactoryMeta): The meta to set.
 |
 |  ----------------------------------------------------------------------
 |  Readonly properties inherited from spacy.language.Language:
 |
 |  component_names
 |      Get the names of the available pipeline components. Includes all
 |      active and inactive pipeline components.
 |
 |      RETURNS (List[str]): List of component name strings, in order.
 |
 |  components
 |      Get all (name, component) tuples in the pipeline, including the
 |      currently disabled components.
 |
 |  disabled
 |      Get the names of all disabled components.
 |
 |      RETURNS (List[str]): The disabled components.
 |
 |  factory_names
 |      Get names of all available factories.
 |
 |      RETURNS (List[str]): The factory names.
 |
 |  path
 |
 |  pipe_factories
 |      Get the component factories for the available pipeline components.
 |
 |      RETURNS (Dict[str, str]): Factory names, keyed by component names.
 |
 |  pipe_labels
 |      Get the labels set by the pipeline components, if available (if
 |      the component exposes a labels property and the labels are not
 |      hidden).
 |
 |      RETURNS (Dict[str, List[str]]): Labels keyed by component name.
 |
 |  pipe_names
 |      Get names of available active pipeline components.
 |
 |      RETURNS (List[str]): List of component name strings, in order.
 |
 |  pipeline
 |      The processing pipeline consisting of (name, component) tuples. The
 |      components are called on the Doc in order as it passes through the
 |      pipeline.
 |
 |      RETURNS (List[Tuple[str, Callable[[Doc], Doc]]]): The pipeline.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from spacy.language.Language:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  config
 |      Trainable config for the current language instance. Includes the
 |      current pipeline components, as well as default training config.
 |
 |      RETURNS (thinc.api.Config): The config.
 |
 |      DOCS: https://spacy.io/api/language#config
 |
 |  meta
 |      Custom meta data of the language class. If a model is loaded, this
 |      includes details from the model's meta.json.
 |
 |      RETURNS (Dict[str, Any]): The meta.
 |
 |      DOCS: https://spacy.io/api/language#meta
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes inherited from spacy.language.Language:
 |
 |  __annotations__ = {'_factory_meta': typing.Dict[str, ForwardRef('Facto…
```

[52]:
```
help(show_entities(d3))
```

```
Tesla                          ORG                            Companies, agencies,
institutions, etc.
Cricket                        Sports                         None
Football                       Sports                         None
Help on NoneType object:

class NoneType(object)
 |  Methods defined here:
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
```

[53]:
```
help(d2.vocab.strings['ORG'])
```

```
Help on int object:

class int(object)
 |  int([x]) -> integer
 |  int(x, base=10) -> integer
 |
 |  Convert a number or string to an integer, or return 0 if no arguments
 |  are given.  If x is a number, return x.__int__().  For floating point
 |  numbers, this truncates towards zero.
 |
 |  If x is not a number or if base is given, then x must be a string,
 |  bytes, or bytearray instance representing an integer literal in the
 |  given base.  The literal can be preceded by '+' or '-' and be surrounded
 |  by whitespace.  The base defaults to 10.  Valid bases are 0 and 2-36.
 |  Base 0 means to interpret the base from the string as an integer literal.
 |  >>> int('0b100', base=0)
 |  4
 |
 |  Built-in subclasses:
 |      bool
 |
 |  Methods defined here:
 |
 |  __abs__(self, /)
 |      abs(self)
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __and__(self, value, /)
 |      Return self&value.
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __ceil__(…)
 |      Ceiling of an Integral returns itself.
 |
 |  __divmod__(self, value, /)
 |      Return divmod(self, value).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __float__(self, /)
 |      float(self)
 |
 |  __floor__(…)
 |      Flooring an Integral returns itself.
 |
 |  __floordiv__(self, value, /)
 |      Return self//value.
 |
 |  __format__(self, format_spec, /)
 |      Default object formatter.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getnewargs__(self, /)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __index__(self, /)
 |      Return self converted to an integer, if self is suitable for use as an
index into a list.
 |
 |  __int__(self, /)
 |      int(self)
 |
 |  __invert__(self, /)
 |      ~self
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __lshift__(self, value, /)
 |      Return self<<value.
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mod__(self, value, /)
 |      Return self%value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __neg__(self, /)
 |      -self
 |
 |  __or__(self, value, /)
 |      Return self|value.
 |
 |  __pos__(self, /)
 |      +self
 |
 |  __pow__(self, value, mod=None, /)
 |      Return pow(self, value, mod).
 |
 |  __radd__(self, value, /)
 |      Return value+self.
 |
 |  __rand__(self, value, /)
 |      Return value&self.
 |
 |  __rdivmod__(self, value, /)
 |      Return divmod(value, self).
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __rfloordiv__(self, value, /)
 |      Return value//self.
 |
 |  __rlshift__(self, value, /)
 |      Return value<<self.
 |
 |  __rmod__(self, value, /)
 |      Return value%self.
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  __ror__(self, value, /)
 |      Return value|self.
 |
 |  __round__(…)
 |      Rounding an Integral returns itself.
 |      Rounding with an ndigits argument also returns an integer.
 |
 |  __rpow__(self, value, mod=None, /)
 |      Return pow(value, self, mod).
 |
 |  __rrshift__(self, value, /)
 |      Return value>>self.
 |
 |  __rshift__(self, value, /)
 |      Return self>>value.
 |
 |  __rsub__(self, value, /)
 |      Return value-self.
 |
 |  __rtruediv__(self, value, /)
 |      Return value/self.
 |
 |  __rxor__(self, value, /)
 |      Return value^self.
```

```
 |
 |  __sizeof__(self, /)
 |      Returns size in memory, in bytes.
 |
 |  __sub__(self, value, /)
 |      Return self-value.
 |
 |  __truediv__(self, value, /)
 |      Return self/value.
 |
 |  __trunc__(…)
 |      Truncating an Integral returns itself.
 |
 |  __xor__(self, value, /)
 |      Return self^value.
 |
 |  as_integer_ratio(self, /)
 |      Return integer ratio.
 |
 |      Return a pair of integers, whose ratio is exactly equal to the original
int
 |      and with a positive denominator.
 |
 |      >>> (10).as_integer_ratio()
 |      (10, 1)
 |      >>> (-10).as_integer_ratio()
 |      (-10, 1)
 |      >>> (0).as_integer_ratio()
 |      (0, 1)
 |
 |  bit_length(self, /)
 |      Number of bits necessary to represent self in binary.
 |
 |      >>> bin(37)
 |      '0b100101'
 |      >>> (37).bit_length()
 |      6
 |
 |  conjugate(…)
 |      Returns self, the complex conjugate of any int.
 |
 |  to_bytes(self, /, length, byteorder, *, signed=False)
 |      Return an array of bytes representing an integer.
 |
 |      length
 |        Length of bytes object to use.  An OverflowError is raised if the
 |        integer is not representable with the given number of bytes.
 |      byteorder
 |          The byte order used to represent the integer.  If byteorder is 'big',
 |          the most significant byte is at the beginning of the byte array.  If
 |          byteorder is 'little', the most significant byte is at the end of the
 |          byte array.  To request the native byte order of the host system, use
 |          `sys.byteorder' as the byte order value.
 |        signed
 |          Determines whether two's complement is used to represent the integer.
 |          If signed is False and a negative integer is given, an OverflowError
 |          is raised.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  from_bytes(bytes, byteorder, *, signed=False) from builtins.type
 |      Return the integer represented by the given array of bytes.
 |
 |      bytes
 |        Holds the array of bytes to convert.  The argument must either
 |        support the buffer protocol or be an iterable object producing bytes.
 |        Bytes and bytearray are examples of built-in objects that support the
 |        buffer protocol.
 |      byteorder
 |        The byte order used to represent the integer.  If byteorder is 'big',
 |        the most significant byte is at the beginning of the byte array.  If
 |        byteorder is 'little', the most significant byte is at the end of the
 |        byte array.  To request the native byte order of the host system, use
 |        `sys.byteorder' as the byte order value.
 |      signed
 |        Indicates whether two's complement is used to represent the integer.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  denominator
 |      the denominator of a rational number in lowest terms
 |
 |  imag
 |      the imaginary part of a complex number
 |
 |  numerator
 |      the numerator of a rational number in lowest terms
 |
 |  real
 |      the real part of a complex number
```

[54]:
```
help(PhraseMatcher(nlp.vocab))
```

```
Help on PhraseMatcher object:

class PhraseMatcher(builtins.object)
 |  PhraseMatcher(Vocab vocab, attr='ORTH', validate=False)
 |  Efficiently match large terminology lists. While the `Matcher` matches
 |      sequences based on lists of token descriptions, the `PhraseMatcher`
accepts
 |      match patterns in the form of `Doc` objects.
 |
 |      DOCS: https://spacy.io/api/phrasematcher
 |      USAGE: https://spacy.io/usage/rule-based-matching#phrasematcher
 |
 |      Adapted from FlashText: https://github.com/vi3k6i5/flashtext
 |      MIT License (see `LICENSE`)
 |      Copyright (c) 2017 Vikash Singh (vikash.duliajan@gmail.com)
 |
 |  Methods defined here:
 |
 |  __call__(…)
 |      Find all sequences matching the supplied patterns on the `Doc`.
 |
 |      doclike (Doc or Span): The document to match over.
 |      as_spans (bool): Return Span objects with labels instead of (match_id,
 |          start, end) tuples.
 |      RETURNS (list): A list of `(match_id, start, end)` tuples,
 |          describing the matches. A match tuple describes a span
 |          `doc[start:end]`. The `match_id` is an integer. If as_spans is set
 |          to True, a list of Span objects is returned.
 |
 |      DOCS: https://spacy.io/api/phrasematcher#call
 |
 |  __contains__(…)
 |      Check whether the matcher contains rules for a match ID.
 |
 |      key (str): The match ID.
 |      RETURNS (bool): Whether the matcher contains rules for this match ID.
 |
 |      DOCS: https://spacy.io/api/phrasematcher#contains
 |
 |  __init__(…)
 |      Initialize the PhraseMatcher.
 |
 |      vocab (Vocab): The shared vocabulary.
 |      attr (int / str): Token attribute to match on.
 |      validate (bool): Perform additional validation when patterns are added.
 |
 |      DOCS: https://spacy.io/api/phrasematcher#init
 |
 |  __len__(…)
 |      Get the number of match IDs added to the matcher.
 |
 |      RETURNS (int): The number of rules.
 |
 |      DOCS: https://spacy.io/api/phrasematcher#len
 |
 |  __reduce__(…)
 |      PhraseMatcher.__reduce__(self)
 |
 |  add(…)
 |      PhraseMatcher.add(self, key, docs, *_docs, on_match=None)
 |      Add a match-rule to the phrase-matcher. A match-rule consists of: an ID
 |              key, an on_match callback, and one or more patterns.
 |
 |              Since spaCy v2.2.2, PhraseMatcher.add takes a list of patterns
as the
 |              second argument, with the on_match callback as an optional
keyword
 |              argument.
 |
 |              key (str): The match ID.
 |              docs (list): List of `Doc` objects representing match patterns.
 |              on_match (callable): Callback executed on match.
 |              *_docs (Doc): For backwards compatibility: list of patterns to
add
 |                  as variable arguments. Will be ignored if a list of patterns
is
 |                  provided as the second argument.
 |
 |              DOCS: https://spacy.io/api/phrasematcher#add
 |
 |  pipe(…)
 |      PhraseMatcher.pipe(self, stream, batch_size=1000, return_matches=False,
as_tuples=False)
 |      Match a stream of documents, yielding them in turn. Deprecated as of
 |              spaCy v3.0.
 |
 |  remove(…)
 |      PhraseMatcher.remove(self, key)
 |      Remove a rule from the matcher by match ID. A KeyError is raised if
 |              the key does not exist.
 |
 |              key (str): The match ID.
 |
 |              DOCS: https://spacy.io/api/phrasematcher#remove
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  vocab
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __pyx_vtable__ = <capsule object NULL>
```

[ ]:

```

```