Doc CLASS

SOURCE

A container for accessing linguistic annotations.

A `Doc` is a sequence of `Token` objects. Access sentences and named entities, export annotations to numpy arrays, losslessly serialize to compressed binary strings. The `Doc` object holds an array of `TokenC` structs. The Python-level `Token` and `Span` objects are views of this array, i.e. they don't own the data themselves.

Doc.__init__ METHOD

Construct a `Doc` object. The most common way to get a `Doc` object is via the `nlp` object.

NAME	DESCRIPTION
`vocab`	A storage container for lexical types. **TYPE:** `Vocab`
`words`	A list of strings or integer hash values to add to the document as words. **TYPE:** `Optional[List[Union[str,int]]]`
`spaces`	A list of boolean values indicating whether each word has a subsequent space. Must have the same length as `words`, if specified. Defaults to a sequence of `True`. **TYPE:** `Optional[List[bool]]`
KEYWORD-ONLY	
`user_data`	Optional extra data to attach to the Doc. **TYPE:** `Dict`
`tags` V3.0	A list of strings, of the same length as `words`, to assign as `token.tag` for each word. Defaults to `None`. **TYPE:** `Optional[List[str]]`
`pos` V3.0	A list of strings, of the same length as `words`, to assign as `token.pos` for each word. Defaults to `None`. **TYPE:** `Optional[List[str]]`
`morphs` V3.0	A list of strings, of the same length as `words`, to assign as `token.morph` for each word. Defaults to `None`. **TYPE:** `Optional[List[str]]`
`lemmas` V3.0	A list of strings, of the same length as `words`, to assign as `token.lemma` for each word. Defaults to `None`. **TYPE:** `Optional[List[str]]`
`heads` V3.0	A list of values, of the same length as `words`, to assign as the head for each word. Head indices are the absolute position of the head in the `Doc`. Defaults to `None`. **TYPE:** `Optional[List[int]]`
`deps` V3.0	A list of strings, of the same length as `words`, to assign as `token.dep` for each word. Defaults to `None`. **TYPE:** `Optional[List[str]]`
`sent_starts` V3.0	A list of values, of the same length as `words`, to assign as `token.is_sent_start`. Will be overridden by heads if `heads` is provided. Defaults to `None`. **TYPE:** `Optional[List[Union[bool, int, None]]]`
`ents` V3.0	A list of strings, of the same length of `words`, to assign the token-based IOB tag. Defaults to `None`

	Defaults to `None`. **TYPE:** `Optional[List[str]]`

Doc.__getitem__ METHOD

Get a `Token` object at position `i`, where `i` is an integer. Negative indexing is supported, and follows the usual Python semantics, i.e. `doc[-2]` is `doc[len(doc) - 2]`.

NAME	DESCRIPTION
`i`	The index of the token. **TYPE:** `int`
RETURNS	The token at `doc[i]`. **TYPE:** `Token`

Get a `Span` object, starting at position `start` (token index) and ending at position `end` (token index). For instance, `doc[2:5]` produces a span consisting of tokens 2, 3 and 4. Stepped slices (e.g. `doc[start : end : step]`) are not supported, as `Span` objects must be contiguous (cannot have gaps). You can use negative indices and open-ended ranges, which have their normal Python semantics.

NAME	DESCRIPTION
`start_end`	The slice of the document to get. **TYPE:** `Tuple[int, int]`
RETURNS	The span at `doc[start:end]`. **TYPE:** `Span`

Doc.__iter__ METHOD

Iterate over `Token` objects, from which the annotations can be easily accessed.

This is the main way of accessing `Token` objects, which are the main way annotations are accessed from Python. If faster-than-Python speeds are required, you can instead access the annotations as a numpy array, or access the underlying C data directly from Cython.

NAME	DESCRIPTION
YIELDS	A `Token` object. TYPE: `Token`

Doc.__len__ METHOD

Get the number of tokens in the document.

NAME	DESCRIPTION
RETURNS	The number of tokens in the document. TYPE: `int`

Doc.set_extension CLASSMETHOD

Define a custom attribute on the `Doc` which becomes available via `Doc._`. For details, see the documentation on custom attributes.

NAME	DESCRIPTION
`name`	Name of the attribute to set by the extension. For example, `"my_attr"` will be available as `doc._.my_attr`. **TYPE:** `str`
`default`	Optional default value of the attribute if no getter or method is defined. **TYPE:** `Optional[Any]`
`method`	Set a custom method on the object, for example `doc._.compare(other_doc)`. **TYPE:** `Optional[Callable[[Doc, …], Any]]`
`getter`	Getter function that takes the object and returns an attribute value. Is called when the user accesses the `._` attribute. **TYPE:** `Optional[Callable[[Doc], Any]]`
`setter`	Setter function that takes the `Doc` and a value, and modifies the object. Is called when the user writes to the `Doc._` attribute. **TYPE:** `Optional[Callable[[Doc, Any], None]]`
`force`	Force overwriting existing attribute. **TYPE:** `bool`

Doc.get_extension CLASSMETHOD

Look up a previously registered extension by name. Returns a 4-tuple `(default, method, getter, setter)` if the extension is registered. Raises a `KeyError` otherwise.

NAME	DESCRIPTION
`name`	Name of the extension. **TYPE:** `str`
RETURNS	A `(default, method, getter, setter)` tuple of the extension. **TYPE:** `Tuple[Optional[Any], Optional[Callable], Optional[Callable], Optional[Callable]]`

Doc.has_extension CLASSMETHOD

Check whether an extension has been registered on the `Doc` class.

NAME	DESCRIPTION
`name`	Name of the extension to check. **TYPE:** `str`
RETURNS	Whether the extension has been registered. **TYPE:** `bool`

Doc.remove_extension CLASSMETHOD

Remove a previously registered extension.

NAME	DESCRIPTION
`name`	Name of the extension. **TYPE:** `str`
RETURNS	A `(default, method, getter, setter)` tuple of the removed extension. **TYPE:** `Tuple[Optional[Any], Optional[Callable], Optional[Callable], Optional[Callable]]`

Doc.char_span METHOD

Create a `Span` object from the slice `doc.text[start_idx:end_idx]`. Returns `None` if the character indices don't map to a valid span using the default alignment mode `"strict".

NAME	DESCRIPTION
`start`	The index of the first character of the span. **TYPE:** `int`
`end`	The index of the last character after the span. **TYPE:** `int`
`label`	A label to attach to the span, e.g. for named entities. **TYPE:** `Union[int, str]`
`kb_id`	An ID from a knowledge base to capture the meaning of a named entity. **TYPE:** `Union[int, str]`
`vector`	A meaning representation of the span. **TYPE:** `numpy.ndarray[ndim=1, dtype=float32]`
`alignment_mode`	How character indices snap to token boundaries. Options: `"strict"` (no snapping), `"contract"` (span of all tokens completely within the character span), `"expand"` (span of all tokens at least partially covered by the character span). Defaults to `"strict"`. **TYPE:** `str`
`span_id` V3.3.1	An identifier to associate with the span. **TYPE:** `Union[int, str]`
RETURNS	The newly constructed object or `None`. **TYPE:** `Optional[Span]`

Doc.set_ents METHOD V3.0

Set the named entities in the document.

NAME	DESCRIPTION
`entities`	Spans with labels to set as entities. **TYPE:** `List[Span]`
KEYWORD-ONLY `blocked`	Spans to set as “blocked” (never an entity) for spacy’s built-in NER component. Other components may ignore this setting. **TYPE:** `Optional[List[Span]]`
`missing`	Spans with missing/unknown entity information. **TYPE:** `Optional[List[Span]]`
`outside`	Spans outside of entities (O in IOB). **TYPE:** `Optional[List[Span]]`
`default`	How to set entity annotation for tokens outside of any provided spans. Options: `"blocked"`, `"missing"`, `"outside"` and `"unmodified"` (preserve current state). Defaults to `"outside"`. **TYPE:** `str`

Doc.similarity METHOD NEEDS MODEL

Make a semantic similarity estimate. The default estimate is cosine similarity using an average of word vectors.

NAME	DESCRIPTION
`other`	The object to compare with. By default, accepts `Doc`, `Span`, `Token` and `Lexeme` objects. **TYPE:** `Union[Doc,Span,Token,Lexeme]`
RETURNS	A scalar similarity score. Higher is more similar. **TYPE:** `float`

Doc.count_by METHOD

Count the frequencies of a given attribute. Produces a dict of `{attr (int): count (ints)}` frequencies, keyed by the values of the given attribute ID.

NAME	DESCRIPTION
`attr_id`	The attribute ID. **TYPE:** `int`
RETURNS	A dictionary mapping attributes to integer counts. **TYPE:** `Dict[int, int]`

Doc.get_lca_matrix METHOD

Calculates the lowest common ancestor matrix for a given `Doc`. Returns LCA matrix containing the integer index of the ancestor, or `-1` if no common ancestor is found, e.g. if span excludes a necessary ancestor.

NAME	DESCRIPTION
RETURNS	The lowest common ancestor matrix of the `Doc`. **TYPE:** `numpy.ndarray[ndim=2, dtype=int32]`

Doc.has_annotation METHOD

Check whether the doc contains annotation on a `Token` attribute .

NAME	DESCRIPTION
`attr`	The attribute string name or int ID. **TYPE:** `Union[int, str]`
KEYWORD-ONLY `require_complete`	Whether to check that the attribute is set on every token in the doc. Defaults to `False`. **TYPE:** `bool`
RETURNS	Whether specified annotation is present in the doc. **TYPE:** `bool`

Doc.to_array METHOD

Export given token attributes to a numpy `ndarray`. If `attr_ids` is a sequence of `M` attributes, the output array will be of shape `(N, M)`, where `N` is the length of the `Doc` (in tokens). If `attr_ids` is a single attribute, the output shape will be `(N,)`. You can specify attributes by integer ID (e.g. `spacy.attrs.LEMMA`) or string name (e.g. “LEMMA” or “lemma”). The values will be 64-bit integers.

Returns a 2D array with one row per token and one column per attribute (when `attr_ids` is a list), or as a 1D numpy array, with one item per attribute (when `attr_ids` is a single value).

NAME	DESCRIPTION
`attr_ids`	A list of attributes (int IDs or string names) or a single attribute (int ID or string name). **TYPE:** `Union[int, str, List[Union[int, str]]]`
RETURNS	The exported attributes as a numpy array. **TYPE:** `Union[numpy.ndarray[ndim=2, dtype=uint64],numpy.ndarray[ndim=1, dtype=uint64]]`

Doc.from_array METHOD

Load attributes from a numpy array. Write to a `Doc` object, from an `(M, N)` array of attributes.

NAME	DESCRIPTION
`attrs`	A list of attribute ID ints. **TYPE:** `List[int]`
`array`	The attribute values to load. **TYPE:** `numpy.ndarray[ndim=2, dtype=int32]`
`exclude`	String names of serialization fields to exclude. **TYPE:** `Iterable[str]`
RETURNS	The `Doc` itself. **TYPE:** `Doc`

Doc.from_docs STATICMETHOD V3.0

Concatenate multiple `Doc` objects to form a new one. Raises an error if the `Doc` objects do not all share the same `Vocab`.

NAME	DESCRIPTION
`docs`	A list of `Doc` objects. **TYPE:** `List[Doc]`
`ensure_whitespace`	Insert a space between two adjacent docs whenever the first doc does not end in whitespace. **TYPE:** `bool`
`attrs`	Optional list of attribute ID ints or attribute name strings. **TYPE:** `Optional[List[Union[str, int]]]`
KEYWORD-ONLY `exclude` V3.3	String names of Doc attributes to exclude. Supported: `spans`, `tensor`, `user_data`. **TYPE:** `Iterable[str]`
RETURNS	The new `Doc` object that is containing the other docs or `None`, if `docs` is empty or `None`. **TYPE:** `Optional[Doc]`

Doc.to_disk METHOD

Save the current state to a directory.

NAME	DESCRIPTION
`path`	A path to a directory, which will be created if it doesn't exist. Paths may be either strings or `Path`-like objects. **TYPE:** `Union[str,Path]`
KEYWORD-ONLY `exclude`	String names of serialization fields to exclude. **TYPE:** `Iterable[str]`

Doc.from_disk METHOD

Loads state from a directory. Modifies the object in place and returns it.

NAME	DESCRIPTION
`path`	A path to a directory. Paths may be either strings or `Path`-like objects. **TYPE:** `Union[str,Path]`
KEYWORD-ONLY `exclude`	String names of serialization fields to exclude. **TYPE:** `Iterable[str]`
RETURNS	The modified `Doc` object. **TYPE:** `Doc`

Doc.to_bytes METHOD

Serialize, i.e. export the document contents to a binary string.

NAME	DESCRIPTION
KEYWORD-ONLY `exclude`	String names of serialization fields to exclude. **TYPE:** `Iterable[str]`
RETURNS	A losslessly serialized copy of the `Doc`, including all annotations. **TYPE:** `bytes`

Doc.from_bytes METHOD

Deserialize, i.e. import the document contents from a binary string.

NAME	DESCRIPTION
`data`	The string to load from. **TYPE:** `bytes`
KEYWORD-ONLY `exclude`	String names of serialization fields to exclude. **TYPE:** `Iterable[str]`
RETURNS	The `Doc` object. **TYPE:** `Doc`

Doc.to_json METHOD

Serializes a document to JSON. Note that this is format differs from the deprecated `JSON training format` .

NAME	DESCRIPTION
`underscore`	Optional list of string names of custom `Doc` attributes. Attribute values need to be JSON-serializable. Values will be added to an `"_"` key in the data, e.g. `"_": {"foo": "bar"}` . **TYPE:** `Optional[List[str]]`
RETURNS	The data in JSON format. **TYPE:** `Dict[str, Any]`

Doc.from_json METHOD V3.3.1

Deserializes a document from JSON, i.e. generates a document from the provided JSON data as generated by `Doc.to_json()` .

NAME	DESCRIPTION
`doc_json`	The Doc data in JSON format from `Doc.to_json`. **TYPE:** `Dict[str, Any]`
KEYWORD-ONLY `validate`	Whether to validate the JSON input against the expected schema for detailed debugging. Defaults to `False` . **TYPE:** `bool`
RETURNS	A `Doc` corresponding to the provided JSON. **TYPE:** `Doc`

Doc.retokenize CONTEXTMANAGER

Context manager to handle retokenization of the `Doc` . Modifications to the `Doc` 's tokenization are stored, and then made all at once when the context manager exits. This is much more efficient, and less error-prone. All views of the `Doc` (`Span` and `Token`) created before the retokenization are invalidated, although they may accidentally continue to work.

NAME	DESCRIPTION
RETURNS	The retokenizer. **TYPE:** `Retokenizer`

Retokenizer.merge METHOD

Mark a span for merging. The `attrs` will be applied to the resulting token (if they're context-dependent token attributes like `LEMMA` or `DEP`) or to the underlying lexeme (if they're context-independent lexical attributes like `LOWER` or `IS_STOP`). Writable custom extension attributes can be provided using the `"_"` key and specifying a dictionary that maps attribute names to values.

NAME	DESCRIPTION
`span`	The span to merge. **TYPE:** `Span`
`attrs`	Attributes to set on the merged token. **TYPE:** `Dict[Union[str, int], Any]`

Retokenizer.split METHOD

Mark a token for splitting, into the specified `orths`. The `heads` are required to specify how the new subtokens should be integrated into the dependency tree. The list of per-token heads can either be a token in the original document, e.g. `doc[2]`, or a tuple consisting of the token in the original document and its subtoken index. For example, `(doc[3], 1)` will attach the subtoken to the second subtoken of `doc[3]`.

This mechanism allows attaching subtokens to other newly created subtokens, without having to keep track of the changing token indices. If the specified head token will be split within the retokenizer block and no subtoken index is specified, it will default to `0`. Attributes to set on subtokens can be provided as a list of values. They'll be applied to the resulting token (if they're context-dependent token attributes like `LEMMA` or `DEP`) or to the underlying lexeme (if they're context-independent lexical attributes like `LOWER` or `IS_STOP`).

NAME	DESCRIPTION
`token`	The token to split. **TYPE:** `Token`
`orths`	The verbatim text of the split tokens. Needs to match the text of the original token. **TYPE:** `List[str]`
`heads`	List of `token` or `(token, subtoken)` tuples specifying the tokens to attach the newly split subtokens to. **TYPE:** `List[Union[Token, Tuple[Token, int]]]`
`attrs`	Attributes to set on all split tokens. Attribute names mapped to list of per-token attribute values. **TYPE:** `Dict[Union[str, int], List[Any]]`

Doc.ents PROPERTY NEEDS MODEL

The named entities in the document. Returns a tuple of named entity `Span` objects, if the entity recognizer has been applied.

NAME	DESCRIPTION
RETURNS	Entities in the document, one `Span` per entity. TYPE: `Tuple[Span]`

Doc.spans PROPERTY

A dictionary of named span groups, to store and access additional span annotations. You can write to it by assigning a list of `Span` objects or a `SpanGroup` to a given key.

NAME	DESCRIPTION
RETURNS	The span groups assigned to the document. TYPE: `Dict[str,SpanGroup]`

Doc.cats PROPERTY NEEDS MODEL

Maps a label to a score for categories applied to the document. Typically set by the `TextCategorizer` .

NAME	DESCRIPTION
RETURNS	The text categories mapped to scores. TYPE: `Dict[str, float]`

Doc.noun_chunks PROPERTY NEEDS MODEL

Iterate over the base noun phrases in the document. Yields base noun-phrase `Span` objects, if the document has been syntactically parsed. A base noun phrase, or "NP chunk", is a noun phrase that does not permit other NPs to be nested within it – so no NP-level coordination, no prepositional phrases, and no relative clauses.

To customize the noun chunk iterator in a loaded pipeline, modify `nlp.vocab.get_noun_chunks` . If the `noun_chunk` syntax iterator has not been implemented for the given language, a `NotImplementedError` is raised.

NAME	DESCRIPTION
YIELDS	Noun chunks in the document. TYPE: `Span`

Doc.sents PROPERTY NEEDS MODEL

Iterate over the sentences in the document. Sentence spans have no label.

This property is only available when sentence boundaries have been set on the document by the `parser`, `senter`, `sentencizer` or some custom function. It will raise an error otherwise.

NAME	DESCRIPTION
YIELDS	Sentences in the document. TYPE: `Span`

Doc.has_vector PROPERTY NEEDS MODEL

A boolean value indicating whether a word vector is associated with the object.

NAME	DESCRIPTION
RETURNS	Whether the document has a vector data attached. TYPE: `bool`

Doc.vector PROPERTY NEEDS MODEL

A real-valued meaning representation. Defaults to an average of the token vectors.

NAME	DESCRIPTION
RETURNS	A 1-dimensional array representing the document's vector. TYPE: `numpy.ndarray[ndim=1, dtype=float32]`

Doc.vector_norm PROPERTY NEEDS MODEL

The L2 norm of the document's vector representation.

NAME	DESCRIPTION
RETURNS	The L2 norm of the vector representation. TYPE: `float`

Attributes

NAME	DESCRIPTION
`text`	A string representation of the document text. **TYPE:** `str`
`text_with_ws`	An alias of `Doc.text`, provided for duck-type compatibility with `Span` and `Token`. **TYPE:** `str`
`mem`	The document's local memory heap, for all C data it owns. **TYPE:** `cymem.Pool`
`vocab`	The store of lexical types. **TYPE:** `Vocab`
`tensor`	Container for dense vector representations. **TYPE:** `numpy.ndarray`
`user_data`	A generic storage area, for user custom data. **TYPE:** `Dict[str, Any]`
`lang`	Language of the document's vocabulary. **TYPE:** `int`
`lang_`	Language of the document's vocabulary. **TYPE:** `str`
`sentiment`	The document's positivity/negativity score, if available. **TYPE:** `float`
`user_hooks`	A dictionary that allows customization of the `Doc`'s properties. **TYPE:** `Dict[str, Callable]`
`user_token_hooks`	A dictionary that allows customization of properties of `Token` children. **TYPE:** `Dict[str, Callable]`
`user_span_hooks`	A dictionary that allows customization of properties of `Span` children. **TYPE:** `Dict[str, Callable]`
`has_unknown_spaces`	Whether the document was constructed without known spacing between tokens (typically when created from gold tokenization). **TYPE:** `bool`
`_`	User space for adding custom attribute extensions.

TYPE: `Underscore`

Serialization fields

During serialization, spaCy will export several data fields used to restore different aspects of the object. If needed, you can exclude them from serialization by passing in the string names via the `exclude` argument.

NAME	DESCRIPTION
`text`	The value of the `Doc.text` attribute.
`sentiment`	The value of the `Doc.sentiment` attribute.
`tensor`	The value of the `Doc.tensor` attribute.
`user_data`	The value of the `Doc.user_data` dictionary.
`user_data_keys`	The keys of the `Doc.user_data` dictionary.
`user_data_values`	The values of the `Doc.user_data` dictionary.

</> SUGGEST EDITS