Fonduer Documentation

Release 0.9.0+dev

HazyResearch

Jun 23, 2021

USER DOCUMENTATION

Fonduer is a Python package and framework for building knowledge base construction (KBC) applications from **richly formatted data**.

Note that Fonduer is still *actively under development*, so feedback and contributions are welcome. Submit bugs in the Issues section or feel free to submit your contributions as a pull request.

CHAPTER

ONE

GETTING STARTED

This document will show you how to get up and running with Fonduer. We'll show you how to get everything installed and your machine so that you can walk through real examples by checking out our Tutorials.

1.1 Installing Non-Python Dependencies

Fonduer relies on a couple of non-Python applications. You'll need to install these and be sure are on your `PATH`.

For OS X using homebrew:

```
$ brew install poppler
$ brew install postgresql@10
$ brew install libpng freetype pkg-config
$ brew install libomp #https://github.com/pytorch/pytorch/issues/20030
$ brew install imagemagick
```

On Debian-based distros:

```
$ sudo apt update
$ sudo apt install libxml2-dev libxslt-dev python3-dev libpq-dev
$ sudo apt build-dep python-matplotlib
$ sudo apt install poppler-utils
$ sudo apt install postgresql
$ sudo apt install libmagickwand-dev
```

Note: Fonduer requires PostgreSQL version 9.6 or higher.

Note: Fonduer requires `poppler-utils` to be version 0.36.0 or later. Otherwise, the `-bbox-layout` option is not available for `pdftotext` (see changelog). It is recommended to use `poppler-utils` version 0.48.0 or later to avoid a known bug.

Note: Use Wand (>=0.5.0) with ImageMagick7 as Wand (<0.5.0) does not support ImageMagick7.

1.2 Installing the Fonduer Package

Then, install Fonduer by running:

```
$ pip install fonduer
```

Note: Fonduer only supports Python 3. Python 2 is not supported.

Tip: For the Python dependencies, we recommend using a virtualenv, which will allow you to install Fonduer and its python dependencies in an isolated Python environment. Once you have virtualenv installed, you can create a Python 3 virtual environment as follows.:

```
$ virtualenv -p python3.6 .venv
```

Once the virtual environment is created, activate it by running:

```
$ source .venv/bin/activate
```

Any Python libraries installed will now be contained within this virtual environment. To deactivate the environment, simply run:

```
$ deactivate
```

1.3 Downloading spaCy language models

Language models introduced recently cannot be downloaded by Fonduer. Those models should be downloaded and their shortcuts should be created as below:

```
$ python -m spacy download ja_core_news_sm
$ python -m spacy link ja_core_news_sm ja
$ python -m spacy download zh_core_web_sm
$ python -m spacy link zh_core_web_sm zh
```

1.4 The Fonduer Pipeline

The Fonduer pipeline can be broken into five phases.

1. **Parsing** In this first stage, an input corpus of richly formatted documents is parsed into Fonduer's data model.
2. **Mention and Candidate Extraction** Here, we initialize the knowledge base with the user's target schema. Users define Mentions using Matchers, and then combine Mentions to create Candidates. Throttlers can also (optionally) be added to filter out invalid Candidates to achieve better class balance.
3. **Multimodal Featurization** Fonduer then featurizes each candidate with features from multiple modalities.
4. **Supervision** Next, users provide labeling functions (which can leverage our data model utilities) to provide weak supervision.
5. **Classification** Finally, Fonduer provides machine learning models which are used to classify each Candidate.

To demonstrate how to set up and use Fonduer in your applications, we walk through each of these phases in real-world examples in our Tutorials.

Check out the Fonduer paper for more details about the system.

CHAPTER

TWO

PARSING

The first stage of Fonduer's pipeline is to parse an input corpus of documents into the Fonduer data model.

Fonduer supports different file formats: CSV/TSV, TXT, HTML, and hOCR. The diagram below illustrates how files in each format are preprocessed and consumed by `Parser`. Nodes in dark blue represent original source files. You have to convert some of them into the formats that Fonduer can consume: a scanned document (incl. non-searchable PDF) is OCRed and exported in hOCR, a (born-digital) PDF is converted into hOCR using tools like pdftotree. It is also possible to convert PDF into HTML using third-party tools, but not recommended (see *Visual Parsers*).

2.1 Multimodal Data Model

The following docs describe elements of Fonduer's data model. These attributes can be used when creating *matchers*, *throttlers*, and *labeling functions*.

Fonduer's parser model module.

class `fonduer.parser.models.`**Caption**(***kwargs*)

Bases: `fonduer.parser.models.context.Context`

A Caption Context in a Document.

Used to represent figure or table captions in a document.

Note: As of v0.6.2, `<caption>` and `<figcaption>` tags turn into `Caption`.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

document
The parent `Document`.

document_id
The id of the parent `Document`.

figure
The parent `Figure`, if any.

figure_id
The id of the parent `Figure`, if any.

id
The unique id the `Caption`.

name
The name of a `Caption`.

position
The position of the `Caption` in the `Document`.

table
The parent `Table`, if any.

table_id
The id of the parent `Table`, if any.

class fonduer.parser.models.**Cell**(***kwargs*)
Bases: fonduer.parser.models.context.Context

A cell Context in a Document.

Used to represent the cells that comprise a table in a document.

Note: As of v0.6.2, `<th>` and `<td>` tags turn into `Cell`.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

col_end
The end index of the column in the `Table` the `Cell` is in.

col_start
The start index of the column in the `Table` the `Cell` is in.

document
The parent `Document`.

document_id
The id of the parent `Document`.

id
The unique id of the `Cell`.

name
The name of a `Cell`.

position
The position of the `Cell` in the `Table`.

row_end
The end index of the row in the `Table` the `Cell` is in.

row_start
The start index of the row in the `Table` the `Cell` is in.

table
The parent `Table`.

table_id
The id of the parent `Table`.

class fonduer.parser.models.**Context**(***kwargs*)
Bases: sqlalchemy.orm.decl_api.Base

A piece of content from which Candidates are composed.

This serves as the base class of the Fonduer document model.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

id
The unique id of the `Context`.

stable_id
A stable representation of the `Context` that will not change between runs.

type
The type of the `Context` represented as a string (e.g. "sentence", "paragraph", "figure").

class fonduer.parser.models.**Document**(***kwargs*)
Bases: fonduer.parser.models.context.Context

A document Context.

Represents all the information of a particular document. What becomes a document depends on which child class of `DocPreprocessor` is used.

Note: As of v0.6.2, each file is one document when `HTMLDocPreprocessor` or `TextDocPreprocessor` is used, each line in the input file is treated as one document when `CSVDocPreprocessor` or `TSVDocPreprocessor` is used.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

id
The unique id of a `Document`.

meta
Pickled metadata about a document extrated from a document preprocessor.

name
The filename of a `Document`, without its extension (e.g., "BC818").

text
The full text of the `Document`.

class fonduer.parser.models.**Figure**(***kwargs*)
Bases: fonduer.parser.models.context.Context

A figure Context in a Document.

Used to represent figures in a document.

Note: As of v0.6.2, <img> and <figure> tags turn into Figure.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

cell
The the parent Cell, if any.

cell_id
The id of the parent Cell, if any.

document
The parent Document.

document_id
The id of the parent Document.

id
The unique id of the Figure.

name
The name of a Figure.

position
The position of the Figure in the Document.

section
The parent Section.

section_id
The id of the parent Section.

url
The Figure's URL.

class fonduer.parser.models.**Paragraph**(***kwargs*)
Bases: fonduer.parser.models.context.Context

A paragraph Context in a Document.

Represents a grouping of adjacent sentences.

Note: As of v0.6.2, a text content in two properties .text and .tail turn into Paragraph. See https://lxml.de/tutorial.html#elements-contain-text for details about .text and .tail properties.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

caption
The parent Caption, if any.

caption_id
The id of the parent Caption, if any.

cell
The parent `Cell`, if any.

cell_id
The id of the parent `Cell`, if any.

document
The parent `Document`.

document_id
The id of the parent `Document`.

id
The unique id of the `Paragraph`.

name
The name of a `Paragraph`.

position
The position of the `Paragraph` in the `Document`.

section
The parent `Section`.

section_id
The id of the parent `Section`.

class fonduer.parser.models.**Section**(***kwargs*)
Bases: fonduer.parser.models.context.Context

A Section Context in a Document.

Note: As of v0.6.2, each document simply has a single Section. Specifically, `<html>` and `<section>` tags turn into `Section`. Future parsing improvements can add better section recognition, such as the sections of an academic paper.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

document
The parent `Document`.

document_id
The id of the parent `Document`.

id
The unique id of the `Section`.

name
The name of a `Section`.

position
The position of the `Section` in a `Document`.

class fonduer.parser.models.**Sentence**(***kwargs*)
Bases: fonduer.parser.models.context.Context, fonduer.parser.models.sentence.TabularMixin, fonduer.parser.models.sentence.LingualMixin, fonduer.

parser.models.sentence.VisualMixin, fonduer.parser.models.sentence.StructuralMixin, fonduer.parser.models.sentence.SentenceMixin

A Sentence subclass with Lingual, Tabular, Visual, and HTML attributes.

Note: Unlike other data models, there is no HTML element corresponding to Sentence. One Paragraph comprises one or more of Sentence, but how a Paragraph is split depends on which NLP parser (e.g., spaCy) is used.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

abs_char_offsets
A list of the character offsets of each word in a Sentence, with respect to the entire document.

bottom
List of each word's BOTTOM bounding box coordinate in the Sentence.

cell
Parent Cell, if any.

cell_id
Id of the parent Cell, if any.

char_offsets
A list of the character offsets of each word in a Sentence, with respect to the start of the sentence.

col_end
col_end of the parent Cell, if any.

col_start
col_start of the parent Cell, if any.

dep_labels
List of dependency labels for each word in a Sentence.

dep_parents
List of the dependency parents for each word in a Sentence.

document
The the parent Document.

document_id
The id of the parent Document.

get_bbox()
Get the bounding box.

Return type Bbox

html_attrs
List of the html attributes of the element containing the Sentence.

html_tag
HTML tag of the element containing the Sentence.

id
The unique id for the Sentence.

is_cellular()
Whether or not the Sentence contains information about its table cell.

Return type bool

is_lingual()
Whether or not the Sentence contains NLP information.

Return type bool

is_structural()
Whether or not the Sentence contains structural information.

Return type bool

is_tabular()
Whether or not the Sentence contains tabular information.

Return type bool

is_visual()
Whether or not the Sentence contains visual information.

Return type bool

left
List of each word's LEFT bounding box coordinate in the Sentence.

lemmas
List of the lemmas for each word in a Sentence.

name
The name of a Sentence.

ner_tags
List of NER tags for each word in a Sentence.

page
List of the page index of each word in the Sentence.

Page indexes start at 1.

paragraph
The parent Paragraph.

paragraph_id
The id of the parent Paragraph.

pos_tags
List of POS tags for each word in a Sentence.

position
The position of the Sentence in the Document.

right
List of each word's RIGHT bounding box coordinate in the Sentence.

row_end
row_end of the parent Cell, if any.

row_start
row_start of the parent Cell, if any.

section
The parent Section.

section_id
The id of the parent `Section`.

table
Parent `Table`, if any.

table_id
Id of the parent `Table`, if any.

text
The full text of the `Sentence`.

top
List of each word's TOP bounding box coordinate in the `Sentence`.

words
A list of the words in a `Sentence`.

xpath
HTML XPATH to the `Sentence`.

class fonduer.parser.models.**Table**(***kwargs*)
Bases: fonduer.parser.models.context.Context

A Table Context in a Document.

Used to represent tables found in a document.

Note: As of v0.6.2, <table> tags turn into `Table`.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

document
The parent `Document`.

document_id
The id of the parent `Document`.

id
The unique id of the `Table`.

name
The name of a `Table`.

position
The position of the `Table` in the `Document`.

section
The parent `Section`.

section_id
The id of the parent `Section`.

class fonduer.parser.models.**Webpage**(***kwargs*)
Bases: fonduer.parser.models.context.Context

A Webpage Context enhanced with additional metadata.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

crawltime
The timestamp of when the `Webpage` was crawled.

host
The host of the `Webpage`.

id
The unique id of the `Webpage`.

name
The name of a `Webpage`.

page_type
The type of the `Webpage`.

raw_content
The raw content of the `Webpage`.

url
The URL of the `Webpage`.

2.2 Core Objects

This is Fonduer's core Parser object.

Fonduer's parser module.

class fonduer.parser.**Parser**(*session*, *parallelism=1*, *structural=True*, *blacklist=['style', 'script']*, *flatten=['span', 'br']*, *language='en'*, *lingual=True*, *lingual_parser=None*, *strip=True*, *replacements=[('[-———]', '-')]*, *tabular=True*, *visual_parser=None*)

Bases: `fonduer.utils.udf.UDFRunner`

Parses into documents into Fonduer's Data Model.

Parameters

- **session** (`Session`) – The database session to use.
- **parallelism** (`int`) – The number of processes to use in parallel. Default 1.
- **structural** (`bool`) – Whether to parse structural information from a DOM.
- **blacklist** (`List[str]`) – A list of tag types to ignore. Default ["style", "script"].
- **flatten** (`List[str]`) – A list of tag types to flatten. Default ["span", "br"]
- **language** (`str`) – Which spaCy NLP language package. Default "en".
- **lingual** (`bool`) – Whether or not to include NLP information. Default True.
- **lingual_parser** (`Optional[LingualParser]`) – A custom lingual parser that inherits *LingualParser*. When specified, *language* will be ignored. When not, `Spacy` with *language* will be used.
- **strip** (`bool`) – Whether or not to strip whitespace during parsing. Default True.

- **replacements** (List[Tuple[str, str]]) – A list of tuples where the regex string in the first position is replaced by the character in the second position. Default [(u"[u2010u2011u2012u2013u2014u2212]", "-")], which replaces various unicode variants of a hyphen (e.g. emdash, endash, minus, etc.) with a standard ASCII hyphen.
- **tabular** (bool) – Whether to include tabular information in the parse.
- **visual_parser** (Optional[VisualParser]) – A visual parser that parses visual information. Defaults to None (visual information is not parsed).

Initialize Parser.

apply(*doc_loader*, *clear=True*, *parallelism=None*, *progress_bar=True*)
Run the Parser.

Parameters

- **doc_loader** (Collection[Document]) – An iteratable of Documents to parse. Typically, one of Fonduer's document preprocessors.
- **clear** (bool) – Whether or not to clear the labels table before applying these LFs.
- **parallelism** (Optional[int]) – How many threads to use for extraction. This will override the parallelism value used to initialize the Labeler if it is provided.
- **progress_bar** (bool) – Whether or not to display a progress bar. The progress bar is measured per document.

Return type None

clear()
Clear all of the Context objects in the database.

Return type None

get_documents()
Return all the successfully parsed Documents in the database.

Return type List[Document]

Returns A list of all Documents in the database ordered by name.

get_last_documents()
Return the most recently successfully parsed list of Documents.

Return type List[Document]

Returns A list of the most recently parsed Documents ordered by name.

last_docs: Set[str]
The last set of documents that apply() was called on

2.3 Lingual Parsers

The following docs describe various lingual parsers. They split text into sentences and enrich them with NLP.

Fonduer's lingual parser module.

class fonduer.parser.lingual_parser.**LingualParser**
Bases: object

Lingual parser.

enrich_sentences_with_NLP(*sentences*)
Add NLP attributes like lemmas, pos_tags, etc. to sentences.

Parameters **sentences** (Collection[Sentence]) – a iterator of *Sentence*.

Return type Iterator[Sentence]

Returns a generator of *Sentence*.

has_NLP_support()
Return True when NLP is supported.

Return type bool

Returns True when NLP is supported.

has_tokenizer_support()
Return True when a tokenizer is supported.

Return type bool

Returns True when a tokenizer is supported.

split_sentences(*text*)
Split input text into sentences.

Parameters **text** (str) – text to be split

Return type Iterable[dict]

Returns A generator of dict that is used as ***kwargs* to instantiate *Sentence*.

class fonduer.parser.lingual_parser.**SimpleParser**(*delim='.'*)
Bases: fonduer.parser.lingual_parser.lingual_parser.LingualParser

Tokenizes text on whitespace only using split().

Parameters **delim** (str) – a delimiter to split text into sentences.

Initialize SimpleParser.

enrich_sentences_with_NLP(*sentences*)
Add NLP attributes like lemmas, pos_tags, etc. to sentences.

Parameters **sentences** (Collection[Sentence]) – a iterator of *Sentence*.

Return type Iterator[Sentence]

Returns a generator of *Sentence*.

has_NLP_support()
Return True when NLP is supported.

Return type bool

Returns True when NLP is supported.

has_tokenizer_support()
Return True when a tokenizer is supported.

Return type bool

Returns True when a tokenizer is supported.

split_sentences(*str*)
Parse the document.

Parameters **str** (str) – The text contents of the document.

Return type Iterator[Dict[str, Any]]

Returns a *generator* of tokenized text.

class fonduer.parser.lingual_parser.**SpacyParser**(*lang*)

Bases: fonduer.parser.lingual_parser.lingual_parser.LingualParser

Spacy parser class.

Parameters **lang** (Optional[str]) – Language. This can be one of ["en", "de", "es", "pt", "fr", "it", "nl", "xx", "ja", "zh"]. See here for details of languages supported by spaCy.

Initialize SpacyParser.

enrich_sentences_with_NLP(*sentences*)

Enrich a list of fonduer Sentence objects with NLP features.

We merge and process the text of all Sentences for higher efficiency.

Parameters **sentences** (Collection[Sentence]) – List of fonduer Sentence objects for one document

Return type Iterator[Sentence]

Returns

has_NLP_support()

Return True when NLP is supported.

Return type bool

Returns True when NLP is supported.

has_tokenizer_support()

Return True when a tokenizer is supported.

Return type bool

Returns True when a tokenizer is supported.

static model_installed(*name*)

Check if spaCy language model is installed.

From https://github.com/explosion/spaCy/blob/master/spacy/util.py

Parameters **name** (str) –

Return type bool

Returns

split_sentences(*text*)

Split text into sentences.

Split input text into sentences that match CoreNLP's default format, but are not yet processed.

Parameters **text** (str) – The text of the parent paragraph of the sentences

Return type Iterator[Dict[str, Any]]

Returns

2.4 Visual Parsers

The following docs describe various visual parsers. They parse visual information, e.g., bounding boxes of each word. Fonduer can parse visual information only for hOCR and HTML files with help of `HocrVisualParser` and `PdfVisualParser`, respectively. It is recommended to provide documents in hOCR instead of HTML, because `PdfVisualParser` is not always accurate by its nature and could assign a wrong bounding box to a word. (see #12). Fonduer's visual parser module.

class fonduer.parser.visual_parser.**HocrVisualParser**(*replacements=[('[--———]', '-')]*)
Bases: fonduer.parser.visual_parser.visual_parser.VisualParser

Visual Parser for hOCR.

Initialize a visual parser.

Raises **ImportError** – an error is raised when spaCy is not 2.3.0 or later.

is_parsable(*document_name*)
Whether visual information can be parsed. Currently always return True.

Parameters **document_name** (str) – the document name.

Return type bool

parse(*document_name*, *sentences*)
Parse visual information embedded in sentence's html_attrs.

Parameters

- **document_name** (str) – the document name.
- **sentences** (Iterable[Sentence]) – sentences to be linked with visual information.

Return type Iterator[Sentence]

Returns A generator of Sentence.

class fonduer.parser.visual_parser.**PdfVisualParser**(*pdf_path*, *verbose=False*)
Bases: fonduer.parser.visual_parser.visual_parser.VisualParser

Link visual information, extracted from PDF, with parsed sentences.

This linker assumes the following conditions for expected results:

- The PDF file exists in a directory specified by *pdf_path*.
- The basename of the PDF file is same as the *document name* and its extension is either ".pdf" or ".PDF".
- A PDF has a text layer.

Initialize VisualParser.

Parameters

- **pdf_path** (str) – a path to directory that contains PDF files.
- **verbose** (bool) – whether to turn on verbose logging.

is_parsable(*document_name*)
Verify that the file exists and has a PDF extension.

Parameters **document_name** (str) – The path to the PDF document.

Return type bool

parse(*document_name*, *sentences*)
Link visual information with sentences.

Parameters

- **document_name** (str) – the document name.
- **sentences** (Iterable[Sentence]) – sentences to be linked with visual information.

Return type Iterator[Sentence]

Returns A generator of Sentence.

class fonduer.parser.visual_parser.**VisualParser**

Bases: abc.ABC

Abstract visual parer.

abstract is_parsable(*document_name*)

Check if visual information can be parsed.

Parameters **document_name** (str) – the document name.

Return type bool

Returns Whether visual information is parsable.

abstract parse(*document_name*, *sentences*)

Parse visual information and link them with given sentences.

Parameters

- **document_name** (str) – the document name.
- **sentences** (Iterable[Sentence]) – sentences to be linked with visual information.

Yield sentences with visual information.

Return type Iterator[Sentence]

2.5 Preprocessors

The following shows descriptions of the various document preprocessors included with Fonduer which are used in parsing documents of different formats.

Fonduer's parser preprocessor module.

class fonduer.parser.preprocessors.**CSVDocPreprocessor**(*path*, *encoding='utf-8'*, *max_docs=9223372036854775807*, *header=False*, *delim=','*, *parser_rule=None*)

Bases: fonduer.parser.preprocessors.doc_preprocessor.DocPreprocessor

A Document generator for CVS files.

It treats each line in the input file as a Document. It assumes that each column is one Section and content in each column as one Paragraph by default. However, if the column is complex, an advanced parser may be used by specifying parser_rule parameter in a dict format where key is the column index and value is the specific parser, e,g., column_constructor in fonduer.utils.utils_parser.

Initialize CSV DocPreprocessor.

Parameters

- **path** (str) – a path to file or directory, or a glob pattern. The basename (as returned by os.path.basename) should be unique among all files.

- **encoding** (str) – file encoding to use (e.g. "utf-8").
- **max_docs** (int) – the maximum number of Documents to produce.
- **header** (bool) – if the CSV file contain header or not, if yes, the header will be used as Section name. default = False
- **delim** (str) – delimiter to be used to separate columns when file has more than one column. It is active only when column is not None. default=','
- **parser_rule** (Optional[Dict[int, Callable]]) – The parser rule to be used to parse the specific column. default = None

Returns A generator of Documents.

class fonduer.parser.preprocessors.**DocPreprocessor**(*path*, *encoding='utf-8'*, *max_docs=9223372036854775807*)

Bases: object

An abstract class of a Document generator.

Unless otherwise stated by a subclass, it's assumed that there is one Document per file.

Initialize DocPreprocessor.

Parameters

- **path** (str) – a path to file or directory, or a glob pattern. The basename (as returned by os.path.basename) should be unique among all files.
- **encoding** (str) – file encoding to use, defaults to "utf-8".
- **max_docs** (int) – the maximum number of Documents to produce, defaults to sys.maxsize.

Returns A generator of Documents.

class fonduer.parser.preprocessors.**HOCRDocPreprocessor**(*path*, *encoding='utf-8'*, *max_docs=9223372036854775807*, *space=True*)

Bases: fonduer.parser.preprocessors.doc_preprocessor.DocPreprocessor

A Document generator for hOCR files.

hOCR should comply with hOCR v1.2. Note that *ppageno* property of *ocr_page* is optional by hOCR v1.2, but is required by Fonduer.

Initialize HOCRDocPreprocessor.

Parameters

- **path** (str) – a path to file or directory, or a glob pattern. The basename (as returned by os.path.basename) should be unique among all files.
- **encoding** (str) – file encoding to use, defaults to "utf-8".
- **max_docs** (int) – the maximum number of Documents to produce, defaults to sys.maxsize.
- **space** (bool) – boolean value indicating whether each word should have a subsequent space. E.g., English has spaces between words.

Returns A generator of Documents.

class fonduer.parser.preprocessors.**HTMLDocPreprocessor**(*path*, *encoding='utf-8'*, *max_docs=9223372036854775807*)

Bases: fonduer.parser.preprocessors.doc_preprocessor.DocPreprocessor

A Document generator for HTML files.

Initialize DocPreprocessor.

Parameters

- **path** (str) – a path to file or directory, or a glob pattern. The basename (as returned by os.path.basename) should be unique among all files.
- **encoding** (str) – file encoding to use, defaults to "utf-8".
- **max_docs** (int) – the maximum number of Documents to produce, defaults to sys.maxsize.

Returns A generator of Documents.

class fonduer.parser.preprocessors.**TSVDocPreprocessor**(*path*, *encoding='utf-8'*, *max_docs=9223372036854775807*, *header=False*)

Bases: fonduer.parser.preprocessors.doc_preprocessor.DocPreprocessor

A Document generator for TSV files.

It treats each line in the input file as a Document. The TSV file should have one (doc_name <tab> doc_text) per line.

Initialize TSV DocPreprocessor.

Parameters

- **path** (str) – a path to file or directory, or a glob pattern. The basename (as returned by os.path.basename) should be unique among all files.
- **encoding** (str) – file encoding to use (e.g. "utf-8").
- **max_docs** (int) – the maximum number of Documents to produce.
- **header** (bool) – if the TSV file contain header or not. default = False

Returns A generator of Documents.

class fonduer.parser.preprocessors.**TextDocPreprocessor**(*path*, *encoding='utf-8'*, *max_docs=9223372036854775807*)

Bases: fonduer.parser.preprocessors.doc_preprocessor.DocPreprocessor

A Document generator for plain text files.

Initialize DocPreprocessor.

Parameters

- **path** (str) – a path to file or directory, or a glob pattern. The basename (as returned by os.path.basename) should be unique among all files.
- **encoding** (str) – file encoding to use, defaults to "utf-8".
- **max_docs** (int) – the maximum number of Documents to produce, defaults to sys.maxsize.

Returns A generator of Documents.

CHAPTER

THREE

DATA MODEL UTILITIES

This page shows descriptions of the utility functions included with Fonduer which can be used to label candidates based on textual, structural, tabular, and visual information. We group each data model utility based on the modality of information that they leverage.

3.1 General Data Model Utilities

Fonduer data model utils.

fonduer.utils.data_model_utils.utils.**get_matches**(*lf*, *candidate_set*, *match_values=[1, -1]*)

Return a list of candidates that are matched by a particular LF.

A simple helper function to see how many matches (non-zero by default) an LF gets.

Parameters

- **lf** (Callable) – The labeling function to apply to the candidate_set
- **candidate_set** (Set[Candidate]) – The set of candidates to evaluate
- **match_values** (List[int]) – An option list of the values to consider as matched. [1, -1] by default.

Return type List[Candidate]

fonduer.utils.data_model_utils.utils.**is_superset**(*a*, *b*)

Check if a is a superset of b.

This is typically used to check if ALL of a list of sentences is in the ngrams returned by an lf_helper.

Parameters

- **a** (Iterable) – A collection of items
- **b** (Iterable) – A collection of items

Return type bool

fonduer.utils.data_model_utils.utils.**overlap**(*a*, *b*)

Check if a overlaps b.

This is typically used to check if ANY of a list of sentences is in the ngrams returned by an lf_helper.

Parameters

- **a** (Iterable) – A collection of items

- **b** (Iterable) – A collection of items

Return type bool

3.2 Textual Data Model Utilities

Fonduer textual modality utilities.

fonduer.utils.data_model_utils.textual.**get_between_ngrams**(*c*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)

Return the ngrams *between* two unary Mentions of a binary-Mention Candidate.

Get the ngrams *between* two unary Mentions of a binary-Mention Candidate, where both share the same sentence Context.

Parameters

- **c** (Candidate) – The binary-Mention Candidate to evaluate.
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If 'True', all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.textual.**get_left_ngrams**(*mention*, *window=3*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)

Get the ngrams within a window to the *left* from the sentence Context.

For higher-arity Candidates, defaults to the *first* argument.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate. If a candidate is given, default to its first Mention.
- **window** (int) – The number of tokens to the left of the first argument to return.
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.textual.**get_neighbor_sentence_ngrams**(*mention*, *d=1*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)

Get the ngrams that are in the neighoring Sentences of the given Mention.

Note that if a candidate is passed in, all of its Mentions will be searched.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose neighbor Sentences are being searched
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.textual.**get_right_ngrams**(*mention*, *window=3*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)

Get the ngrams within a window to the *right* from the sentence Context.

For higher-arity Candidates, defaults to the *last* argument.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate. If a candidate is given, default to its last Mention.
- **window** (int) – The number of tokens to the left of the first argument to return
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.textual.**get_sentence_ngrams**(*mention*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)

Get the ngrams that are in the Sentence of the given Mention, not including itself.

Note that if a candidate is passed in, all of its Mentions will be searched.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose Sentence is being searched
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.textual.**same_sentence**(*c*)

Return True if all Mentions in the given candidate are from the same Sentence.

Parameters **c** (Candidate) – The candidate whose Mentions are being compared

Return type bool

3.3 Structural Data Model Utilities

Fonduer structural modality utilities.

fonduer.utils.data_model_utils.structural.**common_ancestor**(*c*)
Return the path to the root that is shared between a multinary-Mention Candidate.

In particular, this is the common path of HTML tags.

Parameters **c** (Tuple[SpanMention, ...]) – The multinary-Mention Candidate to evaluate

Return type List[str]

fonduer.utils.data_model_utils.structural.**get_ancestor_class_names**(*mention*)
Return the HTML classes of the Mention's ancestors.

If a candidate is passed in, only the ancestors of its first Mention are returned.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate

Return type List[str]

fonduer.utils.data_model_utils.structural.**get_ancestor_id_names**(*mention*)
Return the HTML id's of the Mention's ancestors.

If a candidate is passed in, only the ancestors of its first Mention are returned.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate

Return type List[str]

fonduer.utils.data_model_utils.structural.**get_ancestor_tag_names**(*mention*)
Return the HTML tag of the Mention's ancestors.

For example, ['html', 'body', 'p']. If a candidate is passed in, only the ancestors of its first Mention are returned.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate

Return type List[str]

fonduer.utils.data_model_utils.structural.**get_attributes**(*mention*)
Return the HTML attributes of the Mention.

If a candidate is passed in, only the tag of its first Mention is returned.

A sample outout of this function on a Mention in a paragraph tag is [u'style=padding-top: 8pt;padding-left: 20pt;text-indent: 0pt;text-align: left;']

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate

Return type List[str]

Returns list of strings representing HTML attributes

fonduer.utils.data_model_utils.structural.**get_next_sibling_tags**(*mention*)
Return the HTML tag of the Mention's next siblings.

Next siblings are Mentions which are at the same level in the HTML tree as the given mention, but are declared after the given mention. If a candidate is passed in, only the next siblings of its last Mention are considered in the calculation.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate

Return type List[str]

fonduer.utils.data_model_utils.structural.**get_parent_tag**(*mention*)
Return the HTML tag of the Mention's parent.

These may be tags such as 'p', 'h2', 'table', 'div', etc. If a candidate is passed in, only the tag of its first Mention is returned.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate

Return type Optional[str]

fonduer.utils.data_model_utils.structural.**get_prev_sibling_tags**(*mention*)
Return the HTML tag of the Mention's previous siblings.

Previous siblings are Mentions which are at the same level in the HTML tree as the given mention, but are declared before the given mention. If a candidate is passed in, only the previous siblings of its first Mention are considered in the calculation.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate

Return type List[str]

fonduer.utils.data_model_utils.structural.**get_tag**(*mention*)
Return the HTML tag of the Mention.

If a candidate is passed in, only the tag of its first Mention is returned.

These may be tags such as 'p', 'h2', 'table', 'div', etc.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate

Return type str

fonduer.utils.data_model_utils.structural.**lowest_common_ancestor_depth**(*c*)
Return the lowest common ancestor depth.

In particular, return the minimum distance between a multinary-Mention Candidate to their lowest common ancestor.

For example, if the tree looked like this:

```
html
├──<div> Mention 1 </div>
├──table
│    ├──tr
│    │   └──<th> Mention 2 </th>
```

we return 1, the distance from Mention 1 to the html root. Smaller values indicate that two Mentions are close structurally, while larger values indicate that two Mentions are spread far apart structurally in the document.

Parameters **c** (Tuple[SpanMention, ...]) – The multinary-Mention Candidate to evaluate

Return type int

3.4 Tabular Data Model Utilities

Fonduer tabular modality utilities.

fonduer.utils.data_model_utils.tabular.**get_aligned_ngrams**(*mention*, *attrib='words'*, *n_min=1*, *n_max=1*, *spread=[0, 0]*, *lower=True*)

Get the ngrams from all Cells in the same row or column as the given Mention.

Note that if a candidate is passed in, all of its Mentions will be searched. Also note that if the mention is not tabular, nothing will be yielded.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose row and column Cells are being searched
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **spread** (List[int]) – The number of rows/cols above/below/left/right to also consider "aligned".
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.tabular.**get_cell_ngrams**(*mention*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)

Get the ngrams that are in the Cell of the given mention, not including itself.

Note that if a candidate is passed in, all of its Mentions will be searched. Also note that if the mention is not tabular, nothing will be yielded.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose Cell is being searched
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.tabular.**get_col_ngrams**(*mention*, *attrib='words'*, *n_min=1*, *n_max=1*, *spread=[0, 0]*, *lower=True*)

Get the ngrams from all Cells that are in the same column as the given Mention.

Note that if a candidate is passed in, all of its Mentions will be searched. Also note that if the mention is not tabular, nothing will be yielded.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose column Cells are being searched
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **spread** (List[int]) – The number of cols left and right to also consider "aligned".
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.tabular.**get_head_ngrams**(*mention*, *axis=None*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)

Get the ngrams from the cell in the head of the row or column.

More specifically, this returns the ngrams in the leftmost cell in a row and/or the ngrams in the topmost cell in the column, depending on the axis parameter.

Note that if a candidate is passed in, all of its Mentions will be searched. Also note that if the mention is not tabular, nothing will be yielded.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose head Cells are being returned
- **axis** (Optional[str]) – Which axis {'row', 'col'} to search. If None, then both row and col are searched.
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.tabular.**get_max_col_num**(*mention*)

Return the largest column number that a Mention occupies.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate. If a candidate is given, default to its last Mention.

Return type Optional[int]

fonduer.utils.data_model_utils.tabular.**get_max_row_num**(*mention*)

Return the largest row number that a Mention occupies.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate. If a candidate is given, default to its last Mention.

Return type Optional[int]

fonduer.utils.data_model_utils.tabular.**get_min_col_num**(*mention*)

Return the lowest column number that a Mention occupies.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate. If a candidate is given, default to its first Mention.

Return type Optional[int]

fonduer.utils.data_model_utils.tabular.**get_min_row_num**(*mention*)
Return the lowest row number that a Mention occupies.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate. If a candidate is given, default to its first Mention.

Return type Optional[int]

fonduer.utils.data_model_utils.tabular.**get_neighbor_cell_ngrams**(*mention*, *dist=1*, *directions=False*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)
Get ngrams from all neighbor Cells.

Get the ngrams from all Cells that are within a given Cell distance in one direction from the given Mention.

Note that if a candidate is passed in, all of its Mentions will be searched. If *directions=True*`, each ngram will be returned with a direction in {'UP', 'DOWN', 'LEFT', 'RIGHT'}. Also note that if the mention is not tabular, nothing will be yielded.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose neighbor Cells are being searched
- **dist** (int) – The Cell distance within which a neighbor Cell must be to be considered
- **directions** (bool) – A Boolean expressing whether or not to return the direction of each ngram
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[Union[str, Tuple[str, str]]]

Returns a *generator* of ngrams (or (ngram, direction) tuples if directions=True)

fonduer.utils.data_model_utils.tabular.**get_neighbor_sentence_ngrams**(*mention*, *d=1*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)
Get the ngrams that are in the neighoring Sentences of the given Mention.

Note that if a candidate is passed in, all of its Mentions will be searched.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose neighbor Sentences are being searched
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned

- **lower** (bool) – If True, all ngrams will be returned in lower case

Deprecated since version 0.8.3: This will be removed in 0.9.0. Use `textual.get_neighbor_sentence_ngrams()` instead

Return type Iterator[str]

fonduer.utils.data_model_utils.tabular.**get_row_ngrams**(*mention*, *attrib='words'*, *n_min=1*, *n_max=1*, *spread=[0, 0]*, *lower=True*)

Get the ngrams from all Cells that are in the same row as the given Mention.

Note that if a candidate is passed in, all of its Mentions will be searched. Also note that if the mention is not tabular, nothing will be yielded.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose row Cells are being searched
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **spread** (List[int]) – The number of rows above and below to also consider "aligned".
- **lower** (bool) – If True, all ngrams will be returned in lower case

Return type Iterator[str]

fonduer.utils.data_model_utils.tabular.**get_sentence_ngrams**(*mention*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*)

Get the ngrams that are in the Sentence of the given Mention, not including itself.

Note that if a candidate is passed in, all of its Mentions will be searched.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention whose Sentence is being searched
- **attrib** (str) – The token attribute type (e.g. words, lemmas, poses)
- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case

Deprecated since version 0.8.3: This will be removed in 0.9.0. Use `textual.get_sentence_ngrams()` instead

Return type Iterator[str]

fonduer.utils.data_model_utils.tabular.**is_tabular_aligned**(*c*)

Return True if all Mentions in the given candidate are from the same Row or Col.

Parameters **c** (Candidate) – The candidate whose Mentions are being compared

Return type bool

fonduer.utils.data_model_utils.tabular.**same_cell**(*c*)

Return True if all Mentions in the given candidate are from the same Cell.

Parameters **c** (Candidate) – The candidate whose Mentions are being compared

Return type bool

fonduer.utils.data_model_utils.tabular.**same_col**(*c*)
Return True if all Mentions in the given candidate are from the same Col.

Parameters **c** (Candidate) – The candidate whose Mentions are being compared

Return type bool

fonduer.utils.data_model_utils.tabular.**same_row**(*c*)
Return True if all Mentions in the given candidate are from the same Row.

Parameters **c** (Candidate) – The candidate whose Mentions are being compared

Return type bool

fonduer.utils.data_model_utils.tabular.**same_sentence**(*c*)
Return True if all Mentions in the given candidate are from the same Sentence.

Parameters **c** (Candidate) – The candidate whose Mentions are being compared

Deprecated since version 0.8.3: This will be removed in 0.9.0. Use `textual.same_sentence()` instead

Return type bool

fonduer.utils.data_model_utils.tabular.**same_table**(*c*)
Return True if all Mentions in the given candidate are from the same Table.

Parameters **c** (Candidate) – The candidate whose Mentions are being compared

Return type bool

3.5 Visual Data Model Utilities

Fonduer visual modality utilities.

fonduer.utils.data_model_utils.visual.**get_aligned_lemmas**(*mention*)
Return a set of the lemmas aligned visually with the Mention.

Note that if a candidate is passed in, all of its Mentions will be searched.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate.

Return type Set[str]

fonduer.utils.data_model_utils.visual.**get_horz_ngrams**(*mention*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*, *from_sentence=True*)
Return all ngrams which are visually horizontally aligned with the Mention.

Note that if a candidate is passed in, all of its Mentions will be searched.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate
- **attrib** (str) – The token attribute type (e.g. words, lemmas, pos_tags). This option is valid only when `from_sentence==True`.

- **n_min** (int) – The minimum n of the ngrams that should be returned
- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case
- **from_sentence** (bool) – If True, return ngrams of any Sentence that is horizontally aligned (in the same page) with the mention's Sentence. If False, return ngrams that are horizontally aligned with the mention no matter which Sentence they are from.

Return type Iterator[str]

Returns a *generator* of ngrams

fonduer.utils.data_model_utils.visual.**get_page**(*mention*)

Return the page number of the given mention.

If a candidate is passed in, this returns the page of its first Mention.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to get the page number of.

Return type int

fonduer.utils.data_model_utils.visual.**get_page_horz_percentile**(*mention*, *page_width=612*, *page_height=792*)

Return which percentile from the LEFT in the page the Mention is located in.

Percentile is calculated where the left of the page is 0.0, and the right of the page is 1.0.

Page width and height are based on pt values:

```
Letter      612x792
Tabloid     792x1224
Ledger      1224x792
Legal       612x1008
Statement   396x612
Executive   540x720
A0          2384x3371
A1          1685x2384
A2          1190x1684
A3          842x1190
A4          595x842
A4Small     595x842
A5          420x595
B4          729x1032
B5          516x729
Folio       612x936
Quarto      610x780
10x14       720x1008
```

and should match the source documents. Letter size is used by default.

Note that if a candidate is passed in, only the vertical percentile of its first Mention is returned.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate
- **page_width** (int) – The width of the page. Default to Letter paper width.
- **page_height** (int) – The heigh of the page. Default to Letter paper height.

Return type float

fonduer.utils.data_model_utils.visual.**get_page_vert_percentile**(*mention*, *page_width=612*, *page_height=792*)

Return which percentile from the TOP in the page the Mention is located in.

Percentile is calculated where the top of the page is 0.0, and the bottom of the page is 1.0. For example, a Mention in at the top 1/4 of the page will have a percentile of 0.25.

Page width and height are based on pt values:

```
Letter      612x792
Tabloid     792x1224
Ledger      1224x792
Legal       612x1008
Statement   396x612
Executive   540x720
A0          2384x3371
A1          1685x2384
A2          1190x1684
A3          842x1190
A4          595x842
A4Small     595x842
A5          420x595
B4          729x1032
B5          516x729
Folio       612x936
Quarto      610x780
10x14       720x1008
```

and should match the source documents. Letter size is used by default.

Note that if a candidate is passed in, only the vertical percentil of its first Mention is returned.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate
- **page_width** (int) – The width of the page. Default to Letter paper width.
- **page_height** (int) – The heigh of the page. Default to Letter paper height.

Return type float

fonduer.utils.data_model_utils.visual.**get_vert_ngrams**(*mention*, *attrib='words'*, *n_min=1*, *n_max=1*, *lower=True*, *from_sentence=True*)

Return all ngrams which are visually vertically aligned with the Mention.

Note that if a candidate is passed in, all of its Mentions will be searched.

Parameters

- **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate
- **attrib** (str) – The token attribute type (e.g. words, lemmas, pos_tags). This option is valid only when from_sentence==True.
- **n_min** (int) – The minimum n of the ngrams that should be returned

- **n_max** (int) – The maximum n of the ngrams that should be returned
- **lower** (bool) – If True, all ngrams will be returned in lower case
- **from_sentence** (bool) – If True, return ngrams of any Sentence that is vertically aligned (in the same page) with the mention's Sentence. If False, return ngrams that are vertically aligned with the mention no matter which Sentence they are from.

Return type Iterator[str]

Returns a *generator* of ngrams

fonduer.utils.data_model_utils.visual.**get_vert_ngrams_center**(*c*)
Not implemented.

fonduer.utils.data_model_utils.visual.**get_vert_ngrams_left**(*c*)
Not implemented.

fonduer.utils.data_model_utils.visual.**get_vert_ngrams_right**(*c*)
Not implemented.

fonduer.utils.data_model_utils.visual.**get_visual_aligned_lemmas**(*mention*)
Return a generator of the lemmas aligned visually with the Mention.

Note that if a candidate is passed in, all of its Mentions will be searched.

Parameters **mention** (Union[Candidate, Mention, TemporarySpanMention]) – The Mention to evaluate.

Return type Iterator[str]

fonduer.utils.data_model_utils.visual.**get_visual_distance**(*c*, *axis=None*)
Not implemented.

fonduer.utils.data_model_utils.visual.**get_visual_header_ngrams**(*c*, *axis=None*)
Not implemented.

fonduer.utils.data_model_utils.visual.**is_horz_aligned**(*c*)
Return True if all the components of c are horizontally aligned.

Horizontal alignment means that the bounding boxes of each Mention of c shares a similar y-axis value in the visual rendering of the document.

Parameters **c** (Candidate) – The candidate to evaluate

Return type bool

fonduer.utils.data_model_utils.visual.**is_vert_aligned**(*c*)
Return true if all the components of c are vertically aligned.

Vertical alignment means that the bounding boxes of each Mention of c shares a similar x-axis value in the visual rendering of the document.

Parameters **c** (Candidate) – The candidate to evaluate

Return type bool

fonduer.utils.data_model_utils.visual.**is_vert_aligned_center**(*c*)
Return true if all the components are vertically aligned on their center.

Vertical alignment means that the bounding boxes of each Mention of c shares a similar x-axis value in the visual rendering of the document. In this function the similarity of the x-axis value is based on the center of their bounding boxes.

Parameters **c** (Candidate) – The candidate to evaluate

Return type bool

fonduer.utils.data_model_utils.visual.**is_vert_aligned_left**(*c*)
Return true if all components are vertically aligned on their left border.

Vertical alignment means that the bounding boxes of each Mention of c shares a similar x-axis value in the visual rendering of the document. In this function the similarity of the x-axis value is based on the left border of their bounding boxes.

Parameters **c** (Candidate) – The candidate to evaluate

Return type bool

fonduer.utils.data_model_utils.visual.**is_vert_aligned_right**(*c*)
Return true if all components vertically aligned on their right border.

Vertical alignment means that the bounding boxes of each Mention of c shares a similar x-axis value in the visual rendering of the document. In this function the similarity of the x-axis value is based on the right border of their bounding boxes.

Parameters **c** (Candidate) – The candidate to evaluate

Return type bool

fonduer.utils.data_model_utils.visual.**same_page**(*c*)
Return true if all the components of c are on the same page of the document.

Page numbers are based on the PDF rendering of the document. If a PDF file is provided, it is used. Otherwise, if only a HTML/XML document is provided, a PDF is created and then used to determine the page number of a Mention.

Parameters **c** (Candidate) – The candidate to evaluate

Return type bool

CHAPTER

FOUR

CANDIDATE EXTRACTION

The second stage of Fonduer's pipeline is to extract Mentions and Candidates from the data model.

4.1 Candidate Model Classes

The following describes elements of used for Mention and Candidate extraction.

Fonduer's candidate model module.

class fonduer.candidates.models.**Candidate**(***kwargs*)
Bases: sqlalchemy.orm.decl_api.Base

An abstract candidate relation.

New relation types should be defined by calling candidate_subclass(), **not** subclassing this class directly.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

get_mentions()
Get a tuple of the constituent `Mentions` making up this `Candidate`.

Return type Tuple[Mention, ...]

id
The unique id for the `Candidate`.

split
Which split the `Candidate` belongs to. Used to organize train/dev/test.

type
The type for the `Candidate`, which corresponds to the names the user gives to the candidate_subclasses.

class fonduer.candidates.models.**CaptionMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.caption_mention.TemporaryCaptionMention

A caption `Mention`.

Initialize CaptionMention.

caption
The parent `Caption`.

caption_id
The id of the parent Caption.

get_stable_id()
Return a stable id.

Return type str

id
The unique id of the CaptionMention.

class fonduer.candidates.models.**CellMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.cell_mention.TemporaryCellMention

A cell Mention.

Initialize CellMention.

cell
The parent Cell.

cell_id
The id of the parent Cell.

get_stable_id()
Return a stable id.

Return type str

id
The unique id of the CellMention.

class fonduer.candidates.models.**DocumentMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.document_mention.TemporaryDocumentMention

A document Mention.

Initialize DocumentMention.

document
The parent Document.

document_id
The id of the parent Document.

get_stable_id()
Return a stable id.

Return type str

id
The unique id of the DocumentMention.

class fonduer.candidates.models.**FigureMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.figure_mention.TemporaryFigureMention

A figure Mention.

Initialize FigureMention.

figure
The parent Figure.

figure_id
The id of the parent `Figure`.

get_stable_id()
Return a stable id.

Return type `str`

id
The unique id of the `FigureMention`.

class fonduer.candidates.models.**ImplicitSpanMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.implicit_span_mention.TemporaryImplicitSpanMention

A span of characters that may not appear verbatim in the source text.

It is identified by Context id, character-index start and end (inclusive), as well as a key representing what 'expander' function drew the ImplicitSpanMention from an existing SpanMention, and a position (where position=0 corresponds to the first ImplicitSpanMention produced from the expander function).

The character-index start and end point to the segment of text that was expanded to produce the ImplicitSpanMention.

Initialize ImplicitSpanMention.

char_end
The ending character-index of the `ImplicitSpanMention` (inclusive).

char_start
The starting character-index of the `ImplicitSpanMention`.

dep_labels
A list of the dependency labels for each word in the `ImplicitSpanMention`.

dep_parents
A list of the dependency parents for each word in the `ImplicitSpanMention`.

get_attrib_span(*a*, *sep=''*)
Get the span of sentence attribute *a*.

Intuitively, like calling:

```
sep.join(implicit_span.a)
```

Parameters

- **a** (`str`) – The attribute to get a span for.
- **sep** (`str`) – The separator to use for the join, or to be removed from text if a="words".

Return type `str`

Returns The joined tokens, or text if a="words".

get_attrib_tokens(*a='words'*)
Get the tokens of sentence attribute *a*.

Intuitively, like calling:

```
implicit_span.a
```

Parameters **a** (`str`) – The attribute to get tokens for.

Return type List

Returns The tokens of sentence attribute defined by *a* for the span.

get_bbox()
Get the bounding box.

Return type Bbox

get_num_words()
Get the number of words in the span.

Return type int

Returns The number of words in the span (n of the ngrams).

get_span()
Return the text of the Span.

Return type str

Returns The text of the Span.

get_stable_id()
Return a stable id.

Return type str

get_word_end_index()
Get the index of the ending word of the span.

Return type int

Returns The word-index of the last word of the span.

get_word_start_index()
Get the index of the starting word of the span.

Return type int

Returns The word-index of the start of the span.

id
The unique id of the ImplicitSpanMention.

lemmas
A list of the lemmas for each word in the ImplicitSpanMention.

meta
Pickled metadata about the ImplicitSpanMention.

ner_tags
A list of the NER tags for each word in the ImplicitSpanMention.

page
A list of the page number each word in the ImplicitSpanMention.

pos_tags
A list of the POS tags for each word in the ImplicitSpanMention.

position
The position of the ImplicitSpanMention where position=0 is the first ImplicitSpanMention produced by the expander.

sentence
The parent Sentence.

sentence_id
The id of the parent Sentence.

text
The raw text of the ImplicitSpanMention.

words
A list of the words in the ImplicitSpanMention.

class fonduer.candidates.models.**Mention**(***kwargs*)
Bases: sqlalchemy.orm.decl_api.Base

An abstract Mention.

New mention types should be defined by calling mention_subclass(), **not** subclassing this class directly.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

get_contexts()
Get the constituent context making up this mention.

Return type Tuple[Context, ...]

id
The unique id of the Mention.

type
The type for the Mention, which corresponds to the names the user gives to the mention_subclass.

class fonduer.candidates.models.**ParagraphMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.paragraph_mention.TemporaryParagraphMention

A paragraph Mention.

Initialize ParagraphMention.

get_stable_id()
Return a stable id.

Return type str

id
The unique id of the ParagraphMention.

paragraph
The parent Paragraph.

paragraph_id
The id of the parent Paragraph.

class fonduer.candidates.models.**SectionMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.section_mention.TemporarySectionMention

A section Mention.

Initialize SectionMention.

get_stable_id()
Return a stable id.

Return type str

id
The unique id of the SectionMention.

section
The parent Section.

section_id
The id of the parent Section.

class fonduer.candidates.models.**SpanMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.span_mention.TemporarySpanMention

A span of chars, identified by Context ID and char-index start, end (inclusive).

char_offsets are **relative to the Context start**

Initialize SpanMention.

char_end
The ending character-index of the SpanMention (inclusive).

char_start
The starting character-index of the SpanMention.

get_attrib_span(*a*, *sep=''*)
Get the span of sentence attribute *a*.

Intuitively, like calling:

```
sep.join(span.a)
```

Parameters

- **a** (str) – The attribute to get a span for.
- **sep** (str) – The separator to use for the join, or to be removed from text if a="words".

Return type str

Returns The joined tokens, or text if a="words".

get_attrib_tokens(*a='words'*)
Get the tokens of sentence attribute *a*.

Intuitively, like calling:

```
span.a
```

Parameters **a** (str) – The attribute to get tokens for.

Return type List

Returns The tokens of sentence attribute defined by *a* for the span.

get_bbox()
Get the bounding box.

Return type Bbox

get_num_words()
Get the number of words in the span.

Return type int

Returns The number of words in the span (n of the ngrams).

get_span()
Return the text of the Span.

Return type str

Returns The text of the Span.

get_stable_id()
Return a stable id.

Return type str

get_word_end_index()
Get the index of the ending word of the span.

Return type int

Returns The word-index of the last word of the span.

get_word_start_index()
Get the index of the starting word of the span.

Return type int

Returns The word-index of the start of the span.

id
The unique id of the SpanMention.

meta
Pickled metadata about the ImplicitSpanMention.

sentence
The parent Sentence.

sentence_id
The id of the parent Sentence.

class fonduer.candidates.models.**TableMention**(*tc*)
Bases: fonduer.parser.models.context.Context, fonduer.candidates.models.table_mention.TemporaryTableMention

A table Mention.

Initialize TableMention.

get_stable_id()
Return a stable id.

Return type str

id
The unique id of the TableMention.

table
The parent Table.

table_id
The id of the parent Table.

fonduer.candidates.models.**candidate_subclass**(*class_name*, *args*, *table_name=None*, *cardinality=None*, *values=None*, *nullables=None*)

Create new relation.

Creates and returns a Candidate subclass with provided argument names, which are Context type. Creates the table in DB if does not exist yet.

Import using:

```
from fonduer.candidates.models import candidate_subclass
```

Parameters

- **class_name** (str) – The name of the class, should be "camel case" e.g. NewCandidate
- **args** (List[Mention]) – A list of names of constituent arguments, which refer to the Contexts–representing mentions–that comprise the candidate
- **table_name** (Optional[str]) – The name of the corresponding table in DB; if not provided, is converted from camel case by default, e.g. new_candidate
- **cardinality** (Optional[int]) – The cardinality of the variable corresponding to the Candidate. By default is 2 i.e. is a binary value, e.g. is or is not a true mention.
- **values** (Optional[List[Any]]) – A list of values a candidate can take as their label.
- **nullables** (Optional[List[bool]]) – The number of nullables must match that of args. If nullables[i]==True, a mention for ith mention subclass can be NULL. If nullables=``None`` (by default), no mention can be NULL.

Return type Type[Candidate]

fonduer.candidates.models.**mention_subclass**(*class_name*, *cardinality=None*, *values=None*, *table_name=None*)

Create new mention.

Creates and returns a Mention subclass with provided argument names, which are Context type. Creates the table in DB if does not exist yet.

Import using:

```
from fonduer.candidates.models import mention_subclass
```

Parameters

- **class_name** (str) – The name of the class, should be "camel case" e.g. NewMention
- **table_name** (Optional[str]) – The name of the corresponding table in DB; if not provided, is converted from camel case by default, e.g. new_mention
- **values** (Optional[List[Any]]) – The values that the variable corresponding to the Mention can take. By default it will be [True, False].
- **cardinality** (Optional[int]) – The cardinality of the variable corresponding to the Mention. By default is 2 i.e. is a binary value, e.g. is or is not a true mention.

Return type Type[Mention]

4.2 Core Objects

These are Fonduer's core objects used for Mention and Candidate extraction.

class fonduer.candidates.**MentionExtractor**(*session*, *mention_classes*, *mention_spaces*, *matchers*, *parallelism=1*)

Bases: fonduer.utils.udf.UDFRunner

An operator to extract Mention objects from a Context.

Example Assuming we want to extract two types of Mentions, a Part and a Temperature, and we have already defined Matchers to use:

```
part_ngrams = MentionNgrams(n_max=3)
temp_ngrams = MentionNgrams(n_max=2)

Part = mention_subclass("Part")
Temp = mention_subclass("Temp")

mention_extractor = MentionExtractor(
    session,
    [Part, Temp],
    [part_ngrams, temp_ngrams],
    [part_matcher, temp_matcher]
)
```

Parameters

- **session** (Session) – An initialized database session.
- **mention_classes** (List[Mention]) – The type of relation to extract, defined using :func: fonduer.mentions.mention_subclass.
- **mention_spaces** (List[*MentionSpace*]) – one or list of MentionSpace objects, one for each relation argument. Defines space of Contexts to consider
- **matchers** (List[_Matcher]) – one or list of fonduer.matchers.Matcher objects, one for each relation argument. Only tuples of Contexts for which each element is accepted by the corresponding Matcher will be returned as Mentions
- **parallelism** (int) – The number of processes to use in parallel for calls to apply().

Raises **ValueError** – If mention classes, spaces, and matchers are not the same length.

Initialize the MentionExtractor.

apply(*docs*, *clear=True*, *parallelism=None*, *progress_bar=True*)

Run the MentionExtractor.

Example To extract mentions from a set of training documents using 4 cores:

```
mention_extractor.apply(train_docs, parallelism=4)
```

Parameters

- **docs** (Collection[Document]) – Set of documents to extract from.
- **clear** (bool) – Whether or not to clear the existing Mentions beforehand.
- **parallelism** (Optional[int]) – How many threads to use for extraction. This will override the parallelism value used to initialize the MentionExtractor if it is provided.

- **progress_bar** (bool) – Whether or not to display a progress bar. The progress bar is measured per document.

Return type None

clear()
Delete Mentions of each class in the extractor from the given split.

Return type None

clear_all()
Delete all Mentions from given split the database.

Return type None

get_mentions(*docs=None*, *sort=False*)
Return a list of lists of the mentions associated with this extractor.

Each list of the return will contain the Mentions for one of the mention classes associated with the MentionExtractor.

Parameters

- **docs** (Union[Document, Iterable[Document], None]) – If provided, return Mentions from these documents. Else, return all Mentions.
- **sort** (bool) – If sort is True, then return all Mentions sorted by stable_id.

Return type List[List[Mention]]

Returns Mentions for each mention_class.

last_docs: Set[str]
The last set of documents that apply() was called on

class fonduer.candidates.**CandidateExtractor**(*session*, *candidate_classes*, *throttlers=None*, *self_relations=False*, *nested_relations=False*, *symmetric_relations=True*, *parallelism=1*)

Bases: fonduer.utils.udf.UDFRunner

An operator to extract Candidate objects from a Context.

Example Assuming we have already defined a Part and Temp Mention subclass, and a throttler called templ_throttler, we can create a candidate extractor as follows:

```
PartTemp = candidate_subclass("PartTemp", [Part, Temp])
candidate_extractor = CandidateExtractor(
    session, [PartTemp], throttlers=[temp_throttler]
)
```

Parameters

- **session** (Session) – An initialized database session.
- **candidate_classes** (List[Type[Candidate]]) – The types of relation to extract, defined using fonduer.candidates.candidate_subclass().
- **throttlers** (*list of throttlers.*) – optional functions for filtering out candidates which returns a Boolean expressing whether or not the candidate should be instantiated.
- **self_relations** (bool) – Boolean indicating whether to extract Candidates that relate the same context. Only applies to binary relations.

- **nested_relations** (bool) – Boolean indicating whether to extract Candidates that relate one Context with another that contains it. Only applies to binary relations.
- **symmetric_relations** (bool) – Boolean indicating whether to extract symmetric Candidates, i.e., rel(A,B) and rel(B,A), where A and B are Contexts. Only applies to binary relations.
- **parallelism** (int) – The number of processes to use in parallel for calls to apply().

Raises **ValueError** – If throttlers are provided, but a throtters are not the same length as candidate classes.

Set throttlers match candidate_classes if not provide.

apply(*docs*, *split=0*, *clear=True*, *parallelism=None*, *progress_bar=True*)

Run the CandidateExtractor.

Example To extract candidates from a set of training documents using 4 cores:

```
candidate_extractor.apply(train_docs, split=0, parallelism=4)
```

Parameters

- **docs** (Collection[Document]) – Set of documents to extract from.
- **split** (int) – Which split to assign the extracted Candidates to.
- **clear** (bool) – Whether or not to clear the existing Candidates beforehand.
- **parallelism** (Optional[int]) – How many threads to use for extraction. This will override the parallelism value used to initialize the CandidateExtractor if it is provided.
- **progress_bar** (bool) – Whether or not to display a progress bar. The progress bar is measured per document.

Return type None

clear(*split*)

Clear Candidates of each class.

Delete Candidates of each class initialized with the CandidateExtractor from the given split in the database.

Parameters **split** (int) – Which split to clear.

Return type None

clear_all(*split*)

Delete ALL Candidates from given split the database.

Parameters **split** (int) – Which split to clear.

Return type None

get_candidates(*docs=None*, *split=0*, *sort=False*)

Return a list of lists of the candidates associated with this extractor.

Each list of the return will contain the candidates for one of the candidate classes associated with the CandidateExtractor.

Parameters

- **docs** (Union[Document, Iterable[Document], None]) – If provided, return candidates from these documents from all splits.
- **split** (int) – If docs is None, then return all the candidates from this split.
- **sort** (bool) – If sort is True, then return all candidates sorted by stable_id.

Return type List[List[Candidate]]

Returns Candidates for each candidate_class.

last_docs: Set[str]
The last set of documents that apply() was called on

4.3 MentionSpaces

A *MentionSpace* defines the space of mentions, i.e., the set of all possible mentions. Depending on your needs, you can use a pre-defined child class of MentionSpace or extend one.

class fonduer.candidates.mentions.**MentionSpace**
Bases: object

Define the **space** of Mention objects.

Calling *apply(x)* given an object *x* returns a generator over mentions in *x*.

Initialize mention space.

class fonduer.candidates.mentions.**Ngrams**(*n_min=1*, *n_max=5*, *split_tokens=[]*)
Bases: *fonduer.candidates.mentions.MentionSpace*

Define the space of Mentions as all n-grams in a Sentence.

Define the space of Mentions as all n-grams (n_min <= n <= n_max) in a Sentence *x*, indexing by **character offset**.

Parameters

- **n_min** (int) – Lower limit for the generated n_grams.
- **n_max** (int) – Upper limit for the generated n_grams.
- **split_tokens** (*tuple, list of str.*) – Tokens, on which unigrams are split into two separate unigrams.

Initialize Ngrams.

class fonduer.candidates.mentions.**MentionNgrams**(*n_min=1*, *n_max=5*, *split_tokens=[]*)
Bases: *fonduer.candidates.mentions.Ngrams*

Defines the **space** of Mentions as n-grams in a Document.

Defines the space of Mentions as all n-grams (n_min <= n <= n_max) in a Document *x*, divided into Sentences inside of html elements (such as table cells).

Parameters

- **n_min** (int) – Lower limit for the generated n_grams.
- **n_max** (int) – Upper limit for the generated n_grams.
- **split_tokens** (*tuple, list of str.*) – Tokens, on which unigrams are split into two separate unigrams.

Initialize MentionNgrams.

class fonduer.candidates.mentions.**MentionFigures**(*types=None*)
Bases: *fonduer.candidates.mentions.MentionSpace*

Defines the space of Mentions as all figures in a Document *x*.

Parameters **types** (*list, tuple of str*) – If specified, only yield TemporaryFigureMentions whose url ends in one of the specified types. Example: types=["png", "jpg", "jpeg"].

Initialize MentionFigures.

class fonduer.candidates.mentions.**MentionSentences**
Bases: *fonduer.candidates.mentions.MentionSpace*

Defines the space of Mentions as all sentences in a Document *x*.

Initialize MentionSentences.

class fonduer.candidates.mentions.**MentionParagraphs**
Bases: *fonduer.candidates.mentions.MentionSpace*

Defines the space of Mentions as all paragraphs in a Document *x*.

Initialize MentionParagraphs.

class fonduer.candidates.mentions.**MentionCaptions**
Bases: *fonduer.candidates.mentions.MentionSpace*

Defines the space of Mentions as all captions in a Document *x*.

Initialize MentionCaptions.

class fonduer.candidates.mentions.**MentionCells**
Bases: *fonduer.candidates.mentions.MentionSpace*

Defines the space of Mentions as all cells in a Document *x*.

Initialize MentionCells.

class fonduer.candidates.mentions.**MentionTables**
Bases: *fonduer.candidates.mentions.MentionSpace*

Defines the space of Mentions as all tables in a Document *x*.

Initialize MentionTables.

class fonduer.candidates.mentions.**MentionSections**
Bases: *fonduer.candidates.mentions.MentionSpace*

Defines the space of Mentions as all sections in a Document *x*.

Initialize MentionSections.

class fonduer.candidates.mentions.**MentionDocuments**
Bases: *fonduer.candidates.mentions.MentionSpace*

Defines the space of Mentions as a document in a Document *x*.

Initialize MentionDocuments.

4.4 Matchers

This shows the *matchers* included with Fonduer. These matchers can be used alone, or combined together, to define what spans of text should be made into Mentions.

class fonduer.candidates.matchers.**DateMatcher**(**children*, ***kwargs*)
Bases: *fonduer.candidates.matchers.RegexMatchEach*

Match Spans that are dates, as identified by spaCy.

A convenience class for setting up a RegexMatchEach to match spans for which each token was tagged as a date (DATE).

Initialize date matcher.

class fonduer.candidates.matchers.**DictionaryMatch**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._Matcher

Select mention Ngrams that match against a given list *d*.

Parameters

- **d** (*list of str*) – A list of strings representing a dictionary.
- **ignore_case** (*bool*) – Whether to ignore the case when matching. Default True.
- **inverse** (*bool*) – Whether to invert the results (e.g., return those which are not in the list). Default False.
- **stemmer** – Optionally provide a stemmer to preprocess the dictionary. Can be any object which has a `stem(str) -> str` method like `PorterStemmer()`. Default None.

class fonduer.candidates.matchers.**LambdaFunctionFigureMatcher**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._Matcher

Select Figures that return True when fed to a function f.

Parameters **func** (*function*) – The function to evaluate. See *LambdaFunctionMatcher* for details.

class fonduer.candidates.matchers.**LambdaFunctionMatcher**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._Matcher

Select `Ngrams` that return True when fed to a function f.

Parameters

- **func** (*function*) – The function to evaluate with a signature of `f: m -> {True, False}`, where `m` denotes a mention. More precisely, `m` is an instance of child class of `TemporaryContext`, depending on which `MentionSpace` is used. E.g., `TemporarySpanMention` when `MentionNgrams` is used.
- **longest_match_only** (*bool*) – Whether to only return the longest span matched, rather than all spans. Default False.

class fonduer.candidates.matchers.**LocationMatcher**(**children*, ***kwargs*)
Bases: *fonduer.candidates.matchers.RegexMatchEach*

Match Spans that are the names of locations, as identified by spaCy.

A convenience class for setting up a RegexMatchEach to match spans for which each token was tagged as a location (GPE or LOC).

Initialize location matcher.

class fonduer.candidates.matchers.**MiscMatcher**(**children*, ***kwargs*)
Bases: *fonduer.candidates.matchers.RegexMatchEach*

Match Spans that are miscellaneous named entities, as identified by spaCy.

A convenience class for setting up a RegexMatchEach to match spans for which each token was tagged as miscellaneous (MISC).

Initialize miscellaneous matcher.

class fonduer.candidates.matchers.**NumberMatcher**(**children*, ***kwargs*)
Bases: *fonduer.candidates.matchers.RegexMatchEach*

Match Spans that are numbers, as identified by spaCy.

A convenience class for setting up a RegexMatchEach to match spans for which each token was tagged as a number (CARDINAL or QUANTITY).

Initialize number matcher.

class fonduer.candidates.matchers.**OrganizationMatcher**(**children*, ***kwargs*)
Bases: *fonduer.candidates.matchers.RegexMatchEach*

Match Spans that are the names of organizations, as identified by spaCy.

A convenience class for setting up a RegexMatchEach to match spans for which each token was tagged as an organization (NORG or ORG).

Initialize organization matcher.

class fonduer.candidates.matchers.**PersonMatcher**(**children*, ***kwargs*)
Bases: *fonduer.candidates.matchers.RegexMatchEach*

Match Spans that are the names of people, as identified by spaCy.

A convenience class for setting up a RegexMatchEach to match spans for which each token was tagged as a person (PERSON).

Initialize person matcher.

class fonduer.candidates.matchers.**RegexMatchEach**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._RegexMatch

Match regex pattern on **each token**.

Parameters

- **rgx** (*str*) – The RegEx pattern to use.
- **ignore_case** (*bool*) – Whether or not to ignore case in the RegEx. Default True.
- **full_match** (*bool*) – If True, wrap the provided rgx with `(<rgx>)$`. Default True.
- **longest_match_only** (*bool*) – If True, only return the longest match. Default True.

class fonduer.candidates.matchers.**RegexMatchSpan**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._RegexMatch

Match regex pattern on **full concatenated span**.

Parameters

- **rgx** (*str*) – The RegEx pattern to use.
- **ignore_case** (*bool*) – Whether or not to ignore case in the RegEx. Default True.
- **search** (*bool*) – If True, *search* the regex pattern through the concatenated span. If False, try to *match* the regex patten only at its beginning. Default False.
- **full_match** (*bool*) – If True, wrap the provided rgx with `(<rgx>)$`. Default True.
- **longest_match_only** (*bool*) – If True, only return the longest match. Default True. Will be overridden by the parent matcher like *Union* when it is wrapped by *Union*, *Intersect*, or *Inverse*.

4.5 Matcher Operators

These are the operators which can be use to compose *matchers*.

class fonduer.candidates.matchers.**Concat**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._Matcher

Concatenate mentions generated by Matchers.

Select mentions which are the concatenation of adjacent matches from child operators.

Example A concatenation of a NumberMatcher and PersonMatcher could match on a span of text like "10 Obama".

Parameters

- **permutations** (*bool*) – Default False.
- **left_required** (*bool*) – Whether or not to require the left child to match. Default True.
- **right_required** (*bool*) – Whether or not to require the right child to match. Default True.
- **ignore_sep** (*bool*) – Whether or not to ignore the separator. Default True.
- **sep** – If not ignoring the separator, specify which separator to look for. Default sep=" ".

Raises **ValueError** – If Concat is not provided with two child matcher objects.

Note: Currently slices on **word index** and considers concatenation along these divisions only.

---

class fonduer.candidates.matchers.**Intersect**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._Matcher

Take the intersection of mention sets returned by the provided `Matchers`.

Parameters **longest_match_only** (*bool*) – If True, only return the longest match. Default True. Overrides longest_match_only of its child `Matchers`.

class fonduer.candidates.matchers.**Inverse**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._Matcher

Return the opposite result of ifs child `Matcher`.

Raises **ValueError** – If more than one Matcher is provided.

Parameters **longest_match_only** (*bool*) – If True, only return the longest match. Default True. Overrides longest_match_only of its child `Matchers`.

Initialize inverse matcher.

class fonduer.candidates.matchers.**Union**(**children*, ***opts*)
Bases: fonduer.candidates.matchers._Matcher

Take the union of mention sets returned by the provided `Matchers`.

Parameters **longest_match_only** (*bool*) – If True, only return the longest match. Default True. Overrides longest_match_only of its child `Matchers`.

CHAPTER

FIVE

MULTIMODAL FEATURIZATION

The third stage of Fonduer's pipeline is to featurize each Candidate with multimodal features.

5.1 Feature Model Classes

The following describes the Feature element.

Fonduer's feature model module.

class fonduer.features.models.**Feature**(***kwargs*)

Bases: fonduer.utils.models.annotation.AnnotationMixin, sqlalchemy.orm.decl_api.Base

An element of a representation of a Candidate in a feature space.

A Feature's annotation key identifies the definition of the Feature, e.g., a function that implements it or the library name and feature name in an automatic featurization library.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

candidate
Candidate.

candidate_id
Id of the Candidate being annotated.

keys
List of strings of each Key name.

values: sqlalchemy.sql.schema.Column
A list of floating point values for each Key.

class fonduer.features.models.**FeatureKey**(***kwargs*)

Bases: fonduer.utils.models.annotation.AnnotationKeyMixin, sqlalchemy.orm.decl_api.Base

A feature's key that identifies the definition of the Feature.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

candidate_classes
List of strings of each Key name.

name
Name of the Key.

5.2 Core Objects

These are Fonduer's core objects used for featurization.

Fonduer's features module.

class fonduer.features.**FeatureExtractor**(*features=['textual', 'structural', 'tabular', 'visual']*, *customize_feature_funcs=[]*)

Bases: object

A class to extract features from candidates.

Parameters

- **features** (List[str]) – a list of which Fonduer feature types to extract, defaults to ["textual", "structural", "tabular", "visual"]
- **customize_feature_funcs** (Union[Callable[[List[Candidate]], Iterator[Tuple[int, str, int]]], List[Callable[[List[Candidate]], Iterator[Tuple[int, str, int]]]]]) – a list of customized feature extractors where the extractor takes a list of candidates as input and yield tuples of (candidate_id, feature, value), defaults to []

Initialize FeatureExtractor.

extract(*candidates*)
Extract features from candidates.

Parameters **candidates** (Union[List[Candidate], Candidate]) – A list of candidates to extract features from

Return type Iterator[Tuple[int, str, int]]

class fonduer.features.**Featurizer**(*session*, *candidate_classes*, *feature_extractors=<fonduer.features.feature_extractors.FeatureExtractor object>*, *parallelism=1*)

Bases: fonduer.utils.udf.UDFRunner

An operator to add Feature Annotations to Candidates.

Parameters

- **session** (Session) – The database session to use.
- **candidate_classes** (List[Candidate]) – A list of candidate_subclasses to featurize.
- **parallelism** (int) – The number of processes to use in parallel. Default 1.

Initialize the Featurizer.

apply(*docs=None*, *split=0*, *train=False*, *clear=True*, *parallelism=None*, *progress_bar=True*)
Apply features to the specified candidates.

Parameters

- **docs** (Optional[Collection[Document]]) – If provided, apply features to all the candidates in these documents.
- **split** (int) – If docs is None, apply features to the candidates in this particular split.
- **train** (bool) – Whether or not to update the global key set of features and the features of candidates.
- **clear** (bool) – Whether or not to clear the features table before applying features.
- **parallelism** (Optional[int]) – How many threads to use for extraction. This will override the parallelism value used to initialize the Featurizer if it is provided.
- **progress_bar** (bool) – Whether or not to display a progress bar. The progress bar is measured per document.

Return type None

clear(*train=False*, *split=0*)
Delete Features of each class from the database.

Parameters

- **train** (bool) – Whether or not to clear the FeatureKeys
- **split** (int) – Which split of candidates to clear features from.

Return type None

clear_all()
Delete all Features.

Return type None

drop_keys(*keys*, *candidate_classes=None*)
Drop the specified keys from FeatureKeys.

Parameters

- **keys** (Iterable[str]) – A list of FeatureKey names to delete.
- **candidate_classes** (Union[Candidate, Iterable[Candidate], None]) – A list of the Candidates to drop the key for. If None, drops the keys for all candidate classes associated with this Featurizer.

Return type None

get_feature_matrices(*cand_lists*)
Load sparse matrix of Features for each candidate_class.

Parameters **cand_lists** (List[List[Candidate]]) – The candidates to get features for.

Return type List[csr_matrix]

Returns A list of MxN sparse matrix where M are the candidates and N is the features.

get_keys()
Return a list of keys for the Features.

Return type List[FeatureKey]

Returns List of FeatureKeys.

last_docs: Set[str]
The last set of documents that apply() was called on

update (*docs=None*, *split=0*, *parallelism=None*, *progress_bar=True*)
Update the features of the specified candidates.

Parameters

- **docs** (Optional[Collection[Document]]) – If provided, apply features to all the candidates in these documents.
- **split** (int) – If docs is None, apply features to the candidates in this particular split.
- **parallelism** (Optional[int]) – How many threads to use for extraction. This will override the parallelism value used to initialize the Featurizer if it is provided.
- **progress_bar** (bool) – Whether or not to display a progress bar. The progress bar is measured per document.

Return type None

upsert_keys (*keys*, *candidate_classes=None*)
Upsert the specified keys to FeatureKey.

Parameters

- **keys** (Iterable[str]) – A list of FeatureKey names to upsert.
- **candidate_classes** (Union[Candidate, Iterable[Candidate], None]) – A list of the Candidates to upsert the key for. If None, upsert the keys for all candidate classes associated with this Featurizer.

Return type None

5.3 Multimodal features

Fonduer includes a basic multimodal feature library based on its rich data model. In addition, users can provide their own feature extractors to use with their applications.

Fonduer's feature library module.

fonduer.features.feature_libs.**extract_structural_features** (*candidates*)
Extract structural features.

Parameters **candidates** (Union[Candidate, List[Candidate]]) – A list of candidates to extract features from

Return type Iterator[Tuple[int, str, int]]

fonduer.features.feature_libs.**extract_tabular_features** (*candidates*)
Extract tabular features.

Parameters **candidates** (Union[Candidate, List[Candidate]]) – A list of candidates to extract features from

Return type Iterator[Tuple[int, str, int]]

fonduer.features.feature_libs.**extract_textual_features** (*candidates*)
Extract textual features.

Parameters **candidates** (Union[Candidate, List[Candidate]]) – A list of candidates to extract features from

Return type Iterator[Tuple[int, str, int]]

fonduer.features.feature_libs.**extract_visual_features**(*candidates*)
Extract visual features.

Parameters **candidates** (Union[Candidate, List[Candidate]]) – A list of candidates to extract features from

Return type Iterator[Tuple[int, str, int]]

5.4 Configuration Settings

Visit the Configuring Fonduer page to see how to provide configuration parameters to Fonduer via `.fonduer-config.yaml`.

The different featurization parameters are explained in this section:

```
featurization:
  # settings of textual-based features
  textual:
    # settings for window features
    window_feature:
      size: 3
      combinations: True
      isolated: True
    # settings for word window usd to extract features from surrounding words
    word_feature:
      window: 7
  # settings of tabular-based features
  tabular:
    # unary feture settings
    unary_features:
      # type of attributes
      attrib:
        - words
      # number of gram for features extract in cells
      get_cell_ngrams:
        max: 2
      # number of gram for features extract in headers
      get_head_ngrams:
        max: 2
      # number of gram for features extract in rows
      get_row_ngrams:
        max: 2
      # number of gram for features extract in columns
      get_col_ngrams:
        max: 2
    # binary feature settings
    multinary_features:
      # minimal difference in rows to check
      min_row_diff:
        absolute: False
      # minimal difference in cols to check
      min_col_diff:
        absolute: False
```

CHAPTER

SIX

SUPERVISION

The fourth stage of Fonduer's pipeline is to provide weak supervision which can be used to generate a large set of training data.

6.1 Supervision Model Classes

These are the model classes used for supervision in Fonduer.

Fonduer's supervision model module.

class fonduer.supervision.models.**GoldLabel**(***kwargs*)

Bases: fonduer.utils.models.annotation.AnnotationMixin, sqlalchemy.orm.decl_api.Base

Gold label class.

A separate class for labels from human annotators or other gold standards.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

candidate
Candidate.

candidate_id
Id of the `Candidate` being annotated.

keys
List of strings of each Key name.

values: sqlalchemy.sql.schema.Column
A list of integer values for each Key.

class fonduer.supervision.models.**GoldLabelKey**(***kwargs*)

Bases: fonduer.utils.models.annotation.AnnotationKeyMixin, sqlalchemy.orm.decl_api.Base

Gold label key class.

A gold label's key that identifies the annotator of the gold label.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

candidate_classes
List of strings of each Key name.

name
Name of the Key.

class fonduer.supervision.models.**Label**(***kwargs*)
Bases: fonduer.utils.models.annotation.AnnotationMixin, sqlalchemy.orm.decl_api.Base

Label class.

A discrete label associated with a Candidate, indicating a target prediction value.

Labels are used to represent the output of labeling functions. A Label's annotation key identifies the labeling function that provided the Label.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

candidate
`Candidate`.

candidate_id
Id of the `Candidate` being annotated.

keys
List of strings of each Key name.

values: sqlalchemy.sql.schema.Column
A list of integer values for each Key.

class fonduer.supervision.models.**LabelKey**(***kwargs*)
Bases: fonduer.utils.models.annotation.AnnotationKeyMixin, sqlalchemy.orm.decl_api.Base

Label key class.

A label's key that identifies the labeling function.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

candidate_classes
List of strings of each Key name.

name
Name of the Key.

class fonduer.supervision.models.**StableLabel**(***kwargs*)
Bases: sqlalchemy.orm.decl_api.Base

Stable label table.

A special secondary table for preserving labels created by *human annotators* in a stable format that does not cascade, and is independent of the Candidate IDs.

Note: This is currently unused.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

annotator_name
The annotator's name

context_stable_ids
Delimited list of the context stable ids.

split
Which split the label belongs to

6.2 Core Objects

These are Fonduer's core objects used for supervision.

Fonduer's supervision module.

class fonduer.supervision.**Labeler**(*session*, *candidate_classes*, *parallelism=1*)
Bases: fonduer.utils.udf.UDFRunner

An operator to add Label Annotations to Candidates.

Parameters

- **session** (Session) – The database session to use.
- **candidate_classes** (List[Type[Candidate]]) – A list of candidate_subclasses to label.
- **parallelism** (int) – The number of processes to use in parallel. Default 1.

Initialize the Labeler.

apply(*docs=None*, *split=0*, *train=False*, *lfs=None*, *clear=True*, *parallelism=None*, *progress_bar=True*, *table=<class 'fonduer.supervision.models.label.Label'>*)
Apply the labels of the specified candidates based on the provided LFs.

Parameters

- **docs** (Optional[Collection[Document]]) – If provided, apply the LFs to all the candidates in these documents.
- **split** (int) – If docs is None, apply the LFs to the candidates in this particular split.
- **train** (bool) – Whether or not to update the global key set of labels and the labels of candidates.
- **lfs** (Optional[List[List[Callable]]]) – A list of lists of labeling functions to apply. Each list should correspond with the candidate_classes used to initialize the Labeler.
- **clear** (bool) – Whether or not to clear the labels table before applying these LFs.

- **parallelism** (Optional[int]) – How many threads to use for extraction. This will override the parallelism value used to initialize the Labeler if it is provided.
- **progress_bar** (bool) – Whether or not to display a progress bar. The progress bar is measured per document.
- **table** (Table) – A (database) table labels are written to. Takes *Label* (by default) or *GoldLabel*.

Raises **ValueError** – If labeling functions are not provided for each candidate class.

Return type None

clear(*train*, *split*, *lfs=None*, *table=<class 'fonduer.supervision.models.label.Label'>*, ***kwargs*)
Delete Labels of each class from the database.

Parameters

- **train** (bool) – Whether or not to clear the LabelKeys.
- **split** (int) – Which split of candidates to clear labels from.
- **lfs** (Optional[List[List[Callable]]]) – This parameter is ignored.
- **table** (Table) – A (database) table labels are cleared from. Takes *Label* (by default) or *GoldLabel*.

Return type None

clear_all(*table=<class 'fonduer.supervision.models.label.Label'>*)
Delete all Labels.

Parameters **table** (Table) – A (database) table labels are cleared from. Takes *Label* (by default) or *GoldLabel*.

Return type None

drop_keys(*keys*, *candidate_classes=None*)
Drop the specified keys from LabelKeys.

Parameters

- **keys** (Iterable[Union[str, Callable]]) – A list of labeling functions to delete.
- **candidate_classes** (Union[Type[Candidate], List[Type[Candidate]], None]) – A list of the Candidates to drop the key for. If None, drops the keys for all candidate classes associated with this Labeler.

Return type None

get_gold_labels(*cand_lists*, *annotator=None*)
Load dense matrix of GoldLabels for each candidate_class.

Parameters

- **cand_lists** (List[List[Candidate]]) – The candidates to get gold labels for.
- **annotator** (Optional[str]) – A specific annotator key to get labels for. Default None.

Raises **ValueError** – If get_gold_labels is called before gold labels are loaded, the result will contain ABSTAIN values. We raise a ValueError to help indicate this potential mistake to the user.

Return type List[ndarray]

Returns A list of MxN dense matrix where M are the candidates and N is the annotators. If annotator is provided, return a list of Mx1 matrix.

get_keys()
Return a list of keys for the Labels.

Return type List[LabelKey]

Returns List of LabelKeys.

get_label_matrices(*cand_lists*)
Load dense matrix of Labels for each candidate_class.

Parameters **cand_lists** (List[List[Candidate]]) – The candidates to get labels for.

Return type List[ndarray]

Returns A list of MxN dense matrix where M are the candidates and N is the labeling functions.

last_docs: Set[str]
The last set of documents that apply() was called on

update(*docs=None*, *split=0*, *lfs=None*, *parallelism=None*, *progress_bar=True*, *table=<class 'fonduer.supervision.models.label.Label'>*)
Update the labels of the specified candidates based on the provided LFs.

Parameters

- **docs** (Optional[Collection[Document]]) – If provided, apply the updated LFs to all the candidates in these documents.
- **split** (int) – If docs is None, apply the updated LFs to the candidates in this particular split.
- **lfs** (Optional[List[List[Callable]]]) – A list of lists of labeling functions to update. Each list should correspond with the candidate_classes used to initialize the Labeler.
- **parallelism** (Optional[int]) – How many threads to use for extraction. This will override the parallelism value used to initialize the Labeler if it is provided.
- **progress_bar** (bool) – Whether or not to display a progress bar. The progress bar is measured per document.
- **table** (Table) – A (database) table labels are written to. Takes *Label* (by default) or *GoldLabel*.

Return type None

upsert_keys(*keys*, *candidate_classes=None*)
Upsert the specified keys from LabelKeys.

Parameters

- **keys** (Iterable[Union[str, Callable]]) – A list of labeling functions to upsert.
- **candidate_classes** (Union[Type[Candidate], List[Type[Candidate]], None]) – A list of the Candidates to upsert the key for. If None, upsert the keys for all candidate classes associated with this Labeler.

Return type None

CHAPTER

SEVEN

LEARNING

The final stage of Fonduer's pipeline is to use machine learning models to model the noise between supervision sources to generate probabilistic labels as training data, and then classify each Candidate. Rather than maintaining a separate learning engine, we switch to Emmental, a deep learning framework for multi-task learning. Switching to a more general learning framework allows Fonduer to support more applications and multi-task learning. With Emmental, you need do following steps to perform learning:

1. Create task for each relations and EmmentalModel to learn those tasks.
2. Wrap candidates into EmmentalDataLoader for training.
3. Training and inference (prediction).

7.1 Core Learning Objects

These are Fonduer's core objects used for learning. First, we describe how to create Emmental task for each relation.

Customized Emmental task for Fonduer.

`fonduer.learning.task.`**`create_task`**(*task_names*, *n_arities*, *n_features*, *n_classes*, *emb_layer*, *model='LSTM'*, *mode='MTL'*)

Create task from relation(s).

Parameters

- **`task_names`** (`Union[str, List[str]]`) – Relation name(s), If str, only one relation; If List[str], multiple relations.
- **`n_arities`** (`Union[int, List[int]]`) – The arity of each relation.
- **`n_features`** (`int`) – The multimodal feature set size.
- **`n_classes`** (`Union[int, List[int]]`) – Number of classes for each task. (Only support classification task now).
- **`emb_layer`** (`Optional[EmbeddingModule]`) – The embedding layer for LSTM. No need for LogisticRegression model.
- **`model`** (`str`) – Model name (available models: "LSTM", "LogisticRegression"), defaults to "LSTM".
- **`mode`** (`str`) – Learning mode (available modes: "STL", "MTL"), defaults to "MTL".

Return type `List[EmmentalTask]`

`fonduer.learning.task.`**`loss`**(*module_name*, *intermediate_output_dict*, *Y*, *active*)

Define the loss of the task.

Parameters

- **module_name** (str) – The module name to calculate the loss.
- **intermediate_output_dict** (Dict[str, Any]) – The intermediate output dictionary
- **Y** (Tensor) – Ground truth labels.
- **active** (Tensor) – The sample mask.

Return type Tensor

Returns Loss.

fonduer.learning.task.**output**(*module_name*, *intermediate_output_dict*)

Define the output of the task.

Parameters

- **module_name** (str) – The module name to calculate the loss.
- **intermediate_output_dict** (Dict[str, Any]) – The intermediate output dictionary

Return type Tensor

Returns Output tensor.

Then, we describe how to wrap candidates into an EmmentalDataLoader.

Fonduer dataset.

class fonduer.learning.dataset.**FonduerDataset**(*name*, *candidates*, *features*, *word2id*, *labels*, *index=None*)

Bases: emmental.data.

A FonduerDataset class which is inherited from EmmentalDataset.

This class takes list of candidates and corresponding feature matrix as input and wraps them.

Parameters

- **name** (str) – The name of the dataset.
- **candidates** (List[Candidate]) – The list of candidates.
- **features** (csr_matrix) – The corresponding feature matrix.
- **word2id** (Dict) – The name of the dataset.
- **labels** (Union[array, int]) – If np.array, it's the label for all candidates; If int, it's the number of classes of label and we will create placeholder labels (mainly used for inference).
- **labels** – Which candidates to use. If None, use all candidates.

Initialize FonduerDataset.

7.2 Learning Utilities

These utilities can be used during error analysis to provide additional insights.

Fonduer learning utils.

fonduer.learning.utils.**collect_word_counter**(*candidates*)
Collect word counter from candidates.

Parameters **candidates** (Union[List[Candidate], List[List[Candidate]]]) – The candidates used to collect word counter.

Return type Dict[str, int]

Returns The word counter.

fonduer.learning.utils.**confusion_matrix**(*pred*, *gold*)
Return a confusion matrix.

This can be used for both entity-level and mention-level

Parameters

- **pred** (Set) – a set of predicted entities/candidates
- **gold** (Set) – a set of golden entities/candidates

Return type Tuple[Set, Set, Set]

Returns a tuple of TP, FP, and FN

fonduer.learning.utils.**mark**(*l*, *h*, *idx*)
Produce markers based on argument positions.

Parameters

- **l** (int) – sentence position of first word in argument.
- **h** (int) – sentence position of last word in argument.
- **idx** (int) – argument index (1 or 2).

Return type List[Tuple[int, str]]

Returns markers.

fonduer.learning.utils.**mark_sentence**(*s*, *args*)
Insert markers around relation arguments in word sequence.

Parameters

- **s** (List[str]) – list of tokens in sentence.
- **args** (List[Tuple[int, int, int]]) – list of triples (l, h, idx) as per @_mark(...) corresponding to relation arguments

Return type List[str]

Returns The marked sentence.

Example: Then Barack married Michelle. -> Then ~~[[1 Barack 1]]~~ married ~~[[2 Michelle 2]]~~.

fonduer.learning.utils.**mention_to_tokens**(*mention*, *token_type='words'*, *lowercase=False*)
Extract tokens from the mention.

Parameters

- **mention** (Mention) – mention object.
- **token_type** (str) – token type that wants to extract (e.g. words, lemmas, poses).
- **lowercase** (bool) – use lowercase or not.

Return type List[str]

Returns The token list.

fonduer.learning.utils.**save_marginals**(*session*, *X*, *marginals*, *training=True*)

Save marginal probabilities for a set of Candidates to db.

Parameters

- **X** (List[Candidate]) – A list of arbitrary objects with candidate ids accessible via a .id attrib
- **marginals** (Session) – A dense M x K matrix of marginal probabilities, where K is the cardinality of the candidates, OR a M-dim list/array if K=2.
- **training** (bool) – If True, these are training marginals / labels; else they are saved as end model predictions.

Note: The marginals for k=0 are not stored, only for k = 1,. . . ,K

Return type None

7.3 Configuration Settings

Visit the Configuring Fonduer page to see how to provide configuration parameters to Fonduer via `.fonduer-config.yaml`.

The learning parameters of different models are described below:

```
learning:
  # LSTM model
  LSTM:
    # Word embedding dimension size
    emb_dim: 100
    # The number of features in the LSTM hidden state
    hidden_dim: 100
    # Use attention or not (Options: True or False)
    attention: True
    # Dropout parameter
    dropout: 0.1
    # Use bidirectional LSTM or not (Options: True or False)
    bidirectional: True
  # Logistic Regression model
  LogisticRegression:
    # The number of features in the LogisticRegression hidden state
    hidden_dim: 100
    # bias term
    bias: False
```

CHAPTER

EIGHT

PACKAGING

You can package a whole trained Fonduer pipeline model (parsing, extraction, featurization, and classification) and deploy it to a remote place to serve. To this end, we use MLflow Model as a storage format. A packaged Fonduer pipeline model (or simply referred to as a Fonduer model) looks like this:

Listing 1: Directory written by fonduer.packaging.save_model

```
fonduer_model/
├── MLmodel
├── code
│   ├── my_subclasses.py
│   └── my_fonduer_model.py
├── conda.yaml
├── candidate_classes.pkl
├── mention_classes.pkl
└── model.pkl  # the pickled Fonduer pipeline model.
```

Currently, two types of classifiers are supported: `EmmentalModel` (aka discriminative model) and `LabelModel` (aka generative model). The following example shows how to package a Fonduer pipeline model that uses `EmmentalModel` as a classifier.

8.1 Example

First, create a class that inherits `FonduerModel` and implements `_classify()`. You can see fully functional examples of such a class at hardware_fonduer_model.py and my_fonduer_model.py. Then, put this class in a Python module like *my_fonduer_model.py* instead of in a Jupyter notebook or a Python script as this module will be packaged.

Listing 2: my_fonduer_model.py

```
class MyFonduerModel(FonduerModel):
    def _classify(self, doc: Document) -> DataFrame:
        # Assume only one candidate class is used.
        candidate_class = self.candidate_extractor.candidate_classes[0]
        # Get a list of candidates for this candidate_class.
        test_cands = getattr(doc, candidate_class.__tablename__ + "s")
        # Get a list of true predictions out of candidates.
        ...
        true_preds = [test_cands[_] for _ in positive[0]]

        # Load the true predictions into a dataframe.
        df = DataFrame()
        for true_pred in true_preds:
```

(continues on next page)

(continued from previous page)

```
            entity_relation = tuple(m.context.get_span() for m in true_pred.
↪get_mentions())
            df = df.append(
                DataFrame([entity_relation],
                columns=[m.__name__ for m in candidate_class.mentions]
                )
            )
        return df
```

Similarly, put anything that is required for `MentionExtractor` and `CandidateExtractor`, i.e., mention_classes, mention_spaces, matchers, candidate_classes, and throttlers, into another module.

Listing 3: my_subclasses.py

```
from fonduer.candidates.models import mention_subclass
Presidentname = mention_subclass("Presidentname")
Placeofbirth = mention_subclass("Placeofbirth")

mention_classes = [Presidentname, Placeofbirth]
...
mention_spaces = [presname_ngrams, placeofbirth_ngrams]
matchers = [president_name_matcher, place_of_birth_matcher]
candidate_classes = [PresidentnamePlaceofbirth]
throttlers = [my_throttler]
```

Finally, in a Jupyter notebook or a Python script, build and train a pipeline, then save the trained pipeline.

```
>>> from fonduer.parser.preprocessors import HTMLDocPreprocessor
>>> preprocessor = HTMLDocPreprocessor(docs_path)
>>> ...
# Import mention_classes, candidate_classes, etc. from my_subclasses.py
# instead of defining them here.
>>> from my_subclasses import mention_classes, mention_spaces, matchers
>>> mention_extractor = MentionExtractor(session, mention_classes, mention_spaces,␣
↪matchers)
>>> from my_subclasses import candidate_classes, throttlers
>>> candidate_extractor = CandidateExtractor(session, candidate_classes, throttlers)
>>> ...
>>> from my_fonduer_model import MyFonduerModel
>>> from fonduer.packaging import save_model
>>> save_model(
        fonduer_model=MyFonduerModel(),
        path="fonduer_model",
        code_paths=["my_subclasses.py", "my_fonduer_model.py"],
        preprocessor=preprocessor,
        parser=parser,
        mention_extractor=mention_extractor,
        candidate_extractor=candidate_extractor,
        featurizer=featurizer,
        emmental_model=emmental_model,
        word2id=emb_layer.word2id,
    )
```

Remember to list *my_subclasses.py* and *my_fonduer_model.py* in the `code_paths` argument. Other modules can also be listed if they are required during inference. Alternatively, you can manually place arbitrary modules or data under */code* or */data* directory, respectively. For further information about MLflow Model, please see MLflow Model.

8.2 MLflow model for Fonduer

Customized MLflow model for Fonduer.

class fonduer.packaging.fonduer_model.**FonduerModel**(**args*, ***kwargs*)
Bases: mlflow.pyfunc.

A custom MLflow model for Fonduer.

This class is intended to be subclassed.

static convert_features_to_matrix(*features*, *keys*)
Convert features (the output from FeaturizerUDF.apply) into a sparse matrix.

Parameters

- **features** (List[Dict[str, Any]]) – a list of feature mapping (key: key, value=feature).
- **keys** (List[str]) – a list of all keys.

Return type csr_matrix

static convert_labels_to_matrix(*labels*, *keys*)
Convert labels (the output from LabelerUDF.apply) into a dense matrix.

Note that the input labels are 0-indexed ({0, 1, ..., k}), while the output labels are -1-indexed ({-1, 0, ..., k-1}).

Parameters

- **labels** (List[Dict[str, Any]]) – a list of label mapping (key: key, value=label).
- **keys** (List[str]) – a list of all keys.

Return type ndarray

predict(*model_input*)
Take html_path (and pdf_path) as input and return extracted information.

This method is required and its signature is defined by the MLflow's convention. See MLflow for more details.

Parameters **model_input** (DataFrame) – Pandas DataFrame with rows as docs and colums as params. params should include "html_path" and can optionally include "pdf_path".

Return type DataFrame

Returns Pandas DataFrame containing the output from _classify(), which depends on how it is implemented by a subclass.

fonduer.packaging.fonduer_model.**log_model**(*fonduer_model*, *artifact_path*, *preprocessor*, *parser*, *mention_extractor*, *candidate_extractor*, *conda_env=None*, *code_paths=None*, *model_type='emmental'*, *labeler=None*, *lfs=None*, *label_models=None*, *featurizer=None*, *emmental_model=None*, *word2id=None*)
Log a Fonduer model as an MLflow artifact for the current run.

Parameters

- **fonduer_model** (*FonduerModel*) – Fonduer model to be saved.
- **artifact_path** (str) – Run-relative artifact path.

- **preprocessor** (DocPreprocessor) – the doc preprocessor.
- **parser** (Parser) – self-explanatory
- **mention_extractor** (MentionExtractor) – self-explanatory
- **candidate_extractor** (CandidateExtractor) – self-explanatory
- **conda_env** (Union[Dict, str, None]) – Either a dictionary representation of a Conda environment or the path to a Conda environment yaml file.
- **code_paths** (Optional[List[str]]) – A list of local filesystem paths to Python file dependencies, or directories containing file dependencies. These files are prepended to the system path when the model is loaded.
- **model_type** (Optional[str]) – the model type, either "emmental" or "label", defaults to "emmental".
- **labeler** (Optional[Labeler]) – a labeler, defaults to None.
- **lfs** (Optional[List[List[Callable]]]) – a list of list of labeling functions.
- **label_models** (Optional[List[LabelModel]]) – a list of label models, defaults to None.
- **featurizer** (Optional[Featurizer]) – a featurizer, defaults to None.
- **emmental_model** (Optional[EmmentalModel]) – an Emmental model, defaults to None.
- **word2id** (Optional[Dict]) – a word embedding map.

Return type None

fonduer.packaging.fonduer_model.**save_model**(*fonduer_model*, *path*, *preprocessor*, *parser*, *mention_extractor*, *candidate_extractor*, *mlflow_model=mlflow.models.Model*, *conda_env=None*, *code_paths=None*, *model_type='emmental'*, *labeler=None*, *lfs=None*, *label_models=None*, *featurizer=None*, *emmental_model=None*, *word2id=None*)

Save a Fonduer model to a path on the local file system.

Parameters

- **fonduer_model** (*FonduerModel*) – Fonduer model to be saved.
- **path** (str) – the path on the local file system.
- **preprocessor** (DocPreprocessor) – the doc preprocessor.
- **parser** (Parser) – self-explanatory
- **mention_extractor** (MentionExtractor) – self-explanatory
- **candidate_extractor** (CandidateExtractor) – self-explanatory
- **mlflow_model** (Model) – model configuration.
- **conda_env** (Union[Dict, str, None]) – Either a dictionary representation of a Conda environment or the path to a Conda environment yaml file.
- **code_paths** (Optional[List[str]]) – A list of local filesystem paths to Python file dependencies, or directories containing file dependencies. These files are prepended to the system path when the model is loaded.

- **model_type** (Optional[str]) – the model type, either "emmental" or "label", defaults to "emmental".
- **labeler** (Optional[Labeler]) – a labeler, defaults to None.
- **lfs** (Optional[List[List[Callable]]]) – a list of list of labeling functions.
- **label_models** (Optional[List[LabelModel]]) – a list of label models, defaults to None.
- **featurizer** (Optional[Featurizer]) – a featurizer, defaults to None.
- **emmental_model** (Optional[EmmentalModel]) – an Emmental model, defaults to None.
- **word2id** (Optional[Dict]) – a word embedding map.

Return type None

CHAPTER

NINE

CONFIGURING FONDUER

By default, Fonduer looks for `.fonduer-config.yaml` starting from the current working directory, allowing you to have multiple configuration files for different directories or projects. If it's not there, it looks in parent directories. If no file is found, a default configuration will be used.

Fonduer will only ever use one `.fonduer-config.yaml` file. It does not look for multiple files and will not compose configuration settings from different files.

The default `.fonduer-config.yaml` configuration file is shown below:

```
featurization:
  textual:
    window_feature:
      size: 3
      combinations: True
      isolated: True
    word_feature:
      window: 7
  tabular:
    unary_features:
      attrib:
        - words
      get_cell_ngrams:
        max: 2
      get_head_ngrams:
        max: 2
      get_row_ngrams:
        max: 2
      get_col_ngrams:
        max: 2
    multinary_features:
      min_row_diff:
        absolute: False
      min_col_diff:
        absolute: False

learning:
  LSTM:
    emb_dim: 100
    hidden_dim: 100
    attention: True
    dropout: 0.1
    bidirectional: True
  LogisticRegression:
    hidden_dim: 100
    bias: False
```

About how to customize Emmental, please check here: https://emmental.readthedocs.io/en/latest/user/config.html

CHAPTER

TEN

FREQUENTLY ASKED QUESTIONS (FAQS)

Here are a collection of troubleshooting questions we've seen asked. If you run into anything not covered in this section, feel free to open an Issue.

10.1 When I try to createdb, or use psql, I get FATAL: role "<username>" does not exist.

If you just installed PostgreSQL, you probably need to add users. You will need sudo privileges to do this.

We recommend using createuser to define a new PostgreSQL user account:

```
$ sudo -u postgres createuser [options] [username]
```

10.2 How do I connect to PostgreSQL? I'm getting "fe_sendauth no password supplied".

There are four main ways to deal with entering passwords when you connect to your PostgreSQL database:

1. Set the `PGPASSWORD` environment variable `PGPASSWORD=<pass> psql -h <host> -U <user>`
2. Using a .pgpass file to store the password.
3. Setting the users to trust authentication in the pg_hba.conf file. This makes local development easy, but probably isn't suitable for multiuser environments. You can find your hba file location by running:

   ```
   $ sudo -u postgres psql -c "SHOW hba_file;"
   ```

4. Put the username and password in the connection URI: `postgresql://<user>:<pw>@<host>:<port>/<database_name>`

10.3 I'm getting a CalledProcessError for command 'pdftotext -f 1 -l 1 -bbox-layout'?

Are you using Ubuntu 14.04 (or older)? Fonduer requires `poppler-utils` to be version `0.36.0` or greater. Otherwise, the `-bbox-layout` option is not available for `pdftotext` (see changelog).

If you must use Ubuntu 14.04, you can install manually. As an example, to install `0.53.0`:

```
$ sudo apt install build-essential checkinstall
$ wget poppler.freedesktop.org/poppler-0.53.0.tar.xz
$ tar -xf ./poppler-0.53.0.tar.xz
$ cd poppler-0.53.0
$ ./configure
$ make
$ sudo checkinstall
```

We highly recommend using at least Ubuntu 16.04 though, as we haven't done testing on 14.04 or older.

10.4 How can I use use Fonduer for documents in Languages other than English?

If available, Fonduer uses languages supported by spaCy for tokenization and its NLP pipeline (see spacy language support). We also started adding languages with spaCy alpha support for tokenization (see spacy alpha languages). Currently, only Chinese and Japanese are supported.

If you would like to use Fonduer for Japanese documents, you can use `pip install fonduer[spacy_ja]` to install Fonduer with Japanese language support.

If you would like to use Fonduer for Chinese documents, you can use `pip install fonduer[spacy_zh]` to install Fonduer with Chinese language support.

If you would like to use other languages with spaCy alpha support, which are not yet integrated in Fonduer, feel free to submit a Pull Request or open an Issue.

CHAPTER

ELEVEN

CHANGELOG

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning 2.0.0 conventions. The maintainers will create a git tag for each release and increment the version number found in fonduer/_version.py accordingly. We release tagged versions to PyPI automatically using GitHub Actions.

Note: Fonduer is still under active development and APIs may still change rapidly. Until we release v1.0.0, changes in MINOR version indicate backward incompatible changes.

11.1 Unreleased

11.2 0.9.0 - 2021-06-22

11.2.1 Added

- @HiromuHota: Support spaCy v2.3. (#506)
- @HiromuHota: Add `HOCRDocPreprocessor` and `HocrVisualLinker` to support hOCR as input file. (#476) (#519)
- @YasushiMiyata: Add multiline Japanese strings support to `fonduer.parser.visual_parser.hocr_visual_parser`. (#534) (#542)
- @YasushiMiyata: Add commit process immediately after add to *`fonduer.parser.Parser`*. (#494) (#544)

11.2.2 Changed

- @HiromuHota: Renamed `VisualLinker` to `PdfVisualParser`, which assumes the followings: (#518)
  - `pdf_path` should be a directory path, where PDF files exist, and cannot be a file path.
  - The PDF file should have the same basename (`os.path.basename`) as the document. E.g., the PDF file should be either "123.pdf" or "123.PDF" for "123.html".
- @HiromuHota: Changed `Parser`'s signature as follows: (#518)
  - Renamed `vizlink` to `visual_parser`.
  - Removed `pdf_path`. Now this is required only by `PdfVisualParser`.
  - Removed `visual`. Provide `visual_parser` if visual information is to be parsed.

- @YasushiMiyata: Changed `UDFRunner`'s and `UDF`'s data commit process as follows: (#545)
  - Removed `add` process on single-thread in `_apply()` in `UDFRunner`.
  - Added `UDFRunner._add` of `y` on multi-threads to `Parser`, `Labeler` and `Featurizer`.
  - Removed `y` of document parsed result from `out_queue` in `UDF`.

11.2.3 Fixed

- @YasushiMiyata: Fix test code *test_postgres.py::test_cand_gen_cascading_delete*. (#538) (#539)
- @HiromuHota: Process the tail text only after child elements. (#333) (#520)

11.3 0.8.3 - 2020-09-11

11.3.1 Added

- @YasushiMiyata: Add `get_max_row_num()` to `fonduer.utils.data_model_utils.tabular`. (#469) (#480)
- @HiromuHota: Add get_bbox() to `Sentence` and `SpanMention`. (#429)
- @HiromuHota: Add a custom MLflow model that allows you to package a Fonduer model. See here for how to use it. (#259) (#407)
- @HiromuHota: Support spaCy v2.2. (#384) (#432)
- @wajdikhattel: Add multinary candidates. (#455) (#456)
- @HiromuHota: Add `nullables` to `candidate_subclass()` to allow NULL mention in a candidate. (#496) (#497)
- @HiromuHota: Copy textual functions in `data_model_utils.tabular` to `data_model_utils.textual`. (#503) (#505)

11.3.2 Changed

- @YasushiMiyata: Enable *RegexMatchSpan* with concatenates words by sep="(separator)" option. (#270) (#492)
- @HiromuHota: Enabled "Type hints (PEP 484) support for the Sphinx autodoc extension." (#421)
- @HiromuHota: Switched the Cython wrapper for Mecab from mecab-python3 to fugashi. Since the Japanese tokenizer remains the same, there should be no impact on users. (#384) (#432)
- @HiromuHota: Log a stack trace on parsing error for better debug experience. (#478) (#479)
- @HiromuHota: `get_cell_ngrams()` and `get_neighbor_cell_ngrams()` yield nothing when the mention is not tabular. (#471) (#504)

11.3.3 Deprecated

- @HiromuHota: Deprecated `bbox_from_span()` and `bbox_from_sentence()`. (#429)
- @HiromuHota: Deprecated `visualizer.get_box()` in favor of `span.get_bbox()`. (#445) (#446)
- @HiromuHota: Deprecate textual functions in `data_model_utils.tabular`. (#503) (#505)

11.3.4 Fixed

- @senwu: Fix pdf_path cannot be without a trailing slash. (#442) (#459)
- @kaikun213: Fix bug in table range difference calculations. (#420)
- @HiromuHota: mention_extractor.apply with clear=True now works even if it's not the first run. (#424)
- @HiromuHota: Fix `get_horz_ngrams()` and `get_vert_ngrams()` so that they work even when the input mention is not tabular. (#425) (#426)
- @HiromuHota: Fix the order of args to Bbox. (#443) (#444)
- @HiromuHota: Fix the non-deterministic behavior in VisualLinker. (#412) (#458)
- @HiromuHota: Fix an issue that the progress bar shows no progress on preprocessing by executing preprocessing and parsing in parallel. (#439)
- @HiromuHota: Adopt to mlflow>=1.9.0. (#461) (#463)
- @HiromuHota: Correct the entity type for NumberMatcher from "NUMBER" to "CARDINAL". (#473) (#477)
- @HiromuHota: Fix `_get_axis_ngrams()` not to return `None` when the input is not tabular. (#481)
- @HiromuHota: Fix `Visualizer.display_candidates()` not to draw rectangles on wrong pages. (#488)
- @HiromuHota: Persist doc only when no error happens during parsing. (#489) (#490)

11.4 0.8.2 - 2020-04-28

11.4.1 Deprecated

- @HiromuHota: Use of undecorated labeling functions is deprecated and will not be supported as of v0.9.0. Please decorate them with `snorkel.labeling.labeling_function`.

11.4.2 Fixed

- @HiromuHota: Labeling functions can now be decorated with `snorkel.labeling.labeling_function`. (#400) (#401)

11.5 0.8.1 - 2020-04-13

11.5.1 Added

- @senwu: Add *mode* argument in create_task to support *STL* and *MTL*.

Note: Fonduer has a new *mode* argument to support switching between different learning modes (e.g., STL or MLT). Example usage:

```
# Create task for each relation.
tasks = create_task(
    task_names = TASK_NAMES,
    n_arities = N_ARITIES,
    n_features = N_FEATURES,
    n_classes = N_CLASSES,
    emb_layer = EMB_LAYER,
    model="LogisticRegression",
    mode = MODE,
)
```

11.6 0.8.0 - 2020-04-07

11.6.1 Changed

- @senwu: Switch to Emmental as the default learning engine.

Note: Rather than maintaining a separate learning engine, we switch to Emmental, a deep learning framework for multi-task learning. Switching to a more general learning framework allows Fonduer to support more applications and multi-task learning. Example usage:

```
# With Emmental, you need do following steps to perform learning:
# 1. Create task for each relations and EmmentalModel to learn those tasks.
# 2. Wrap candidates into EmmentalDataLoader for training.
# 3. Training and inference (prediction).

import emmental

# Collect word counter from candidates which is used in LSTM model.
word_counter = collect_word_counter(train_cands)

# Initialize Emmental. For customize Emmental, please check here:
# https://emmental.readthedocs.io/en/latest/user/config.html
emmental.init(fonduer.Meta.log_path)

########################################################################
# 1. Create task for each relations and EmmentalModel to learn those tasks.
########################################################################

# Generate special tokens which are used for LSTM model to locate mentions.
# In LSTM model, we pad sentence with special tokens to help LSTM to learn
# those mentions. Example:
```

(continues on next page)

(continued from previous page)

```
# Original sentence: Then Barack married Michelle.
# ->  Then ~~[[1 Barack 1]]~~ married ~~[[2 Michelle 2]]~~.
arity = 2
special_tokens = []
for i in range(arity):
    special_tokens += [f"~~[[{i}", f"{i}]]~~"]

# Generate word embedding module for LSTM.
emb_layer = EmbeddingModule(
    word_counter=word_counter, word_dim=300, specials=special_tokens
)

# Create task for each relation.
tasks = create_task(
    ATTRIBUTE,
    2,
    F_train[0].shape[1],
    2,
    emb_layer,
    model="LogisticRegression",
)

# Create Emmental model to learn the tasks.
model = EmmentalModel(name=f"{ATTRIBUTE}_task")

# Add tasks into model
for task in tasks:
    model.add_task(task)

######################################################################
# 2. Wrap candidates into EmmentalDataLoader for training.
######################################################################

# Here we only use the samples that have labels, which we filter out the
# samples that don't have significant marginals.
diffs = train_marginals.max(axis=1) - train_marginals.min(axis=1)
train_idxs = np.where(diffs > 1e-6)[0]

# Create a dataloader with weakly supervisied samples to learn the model.
train_dataloader = EmmentalDataLoader(
    task_to_label_dict={ATTRIBUTE: "labels"},
    dataset=FonduerDataset(
        ATTRIBUTE,
        train_cands[0],
        F_train[0],
        emb_layer.word2id,
        train_marginals,
        train_idxs,
    ),
    split="train",
    batch_size=100,
    shuffle=True,
)


# Create test dataloader to do prediction.
# Build test dataloader
```

(continues on next page)

(continued from previous page)

```
test_dataloader = EmmentalDataLoader(
    task_to_label_dict={ATTRIBUTE: "labels"},
    dataset=FonduerDataset(
        ATTRIBUTE, test_cands[0], F_test[0], emb_layer.word2id, 2
    ),
    split="test",
    batch_size=100,
    shuffle=False,
)


########################################################################
# 3. Training and inference (prediction).
########################################################################

# Learning those tasks.
emmental_learner = EmmentalLearner()
emmental_learner.learn(model, [train_dataloader])

# Predict based the learned model.
test_preds = model.predict(test_dataloader, return_preds=True)
```

- @HiromuHota: Change ABSTAIN to -1 to be compatible with Snorkel of 0.9.X. Accordingly, user-defined labels should now be 0-indexed (used to be 1-indexed). (#310) (#320)
- @HiromuHota: Use executemany_mode="batch" instead of deprecated use_batch_mode=True. (#358)
- @HiromuHota: Use tqdm.notebook.tqdm instead of deprecated tqdm.tqdm_notebook. (#360)
- @HiromuHota: To support ImageMagick7, expand the version range of Wand. (#373)
- @HiromuHota: Comply with PEP 561 for type-checking codes that use Fonduer.
- @HiromuHota: Make UDF.apply of all child classes unaware of the database backend, meaning PostgreSQL is not required if UDF.apply is directly used instead of UDFRunner.apply. (#316) (#368)

11.6.2 Fixed

- @senwu: Fix mention extraction to return mention classes instead of data model classes.

11.7 0.7.1 - 2019-11-06

11.7.1 Added

- @senwu: Refactor *Featurization* to support user defined customized feature extractors and rename existing feature extractors' name to match the paper.

Note: Rather than using a fixed multimodal feature library along, we have added an interface for users to provide customized feature extractors. Please see our full documentation for details.

```
from fonduer.features import Featurizer, FeatureExtractor
```

(continues on next page)

(continued from previous page)

```
# Example feature extractor
def feat_ext(candidates):
    for candidate in candidates:
        yield candidate.id, f"{candidate.id}", 1

feature_extractors=FeatureExtractor(customize_feature_funcs=[feat_ext])
featurizer = Featurizer(session, [PartTemp], feature_extractors=feature_extractors)
```

Rather than:

```
from fonduer.features import Featurizer

featurizer = Featurizer(session, [PartTemp])
```

- @HiromuHota: Add page argument to get_pdf_dim in case pages have different dimensions.
- @HiromuHota: Add Labeler#upsert_keys.
- @HiromuHota: Add *vizlink* as an argument to *Parser* to be able to plug a custom visual linker. Unless otherwise specified, *VisualLinker* will be used by default.

Note: Example usage:

```
from fonduer.parser.visual_linker import VisualLinker
class CustomVisualLinker(VisualLinker):
    def __init__(self):
        """Your code"""

    def link(self, document_name: str, sentences: Iterable[Sentence], pdf_path: str) -
↪> Iterable[Sentence]:
        """Your code"""

    def is_linkable(self, filename: str) -> bool:
        """Your code"""

from fonduer.parser import Parser
parser = Parser(session, vizlink=CustomVisualLinker())
```

- @HiromuHota: Add *LingualParser*, which any lingual parser like *Spacy* should inherit from, and add *lingual_parser* as an argument to *Parser* to be able to plug a custom lingual parser.
- @HiromuHota: Annotate types to some of the classes incl. preprocesssors and parser/models.
- @HiromuHota: Add table argument to `Labeler.apply` (and `Labeler.update`), which can now be used to annotate gold labels.

Note: Example usage:

```
# Define a LF for gold labels
def gold(c: Candidate) -> int:
    if some condition:
        return TRUE
    else:
        return FALSE
```

(continues on next page)

(continued from previous page)

```
labeler = Labeler(session, [PartTemp, PartVolt])
# Annotate gold labels
labeler.apply(docs=docs, lfs=[[gold], [gold]], table=GoldLabel, train=True)
# A label matrix can be obtained using the name of annotator, "gold" in this case
L_train_gold = labeler.get_gold_labels(train_cands, annotator="gold")
# Annotate (noisy) labels
labeler.apply(split=0, lfs=[[LF1, LF2, LF3], [LF4, LF5]], train=True)
```

Note that the method name, "gold" in this example, is referred to as annotator.

11.7.2 Changed

- @HiromuHota: Load a spaCy model if possible during *Spacy#__init__*.
- @HiromuHota: Rename Spacy to SpacyParser.
- @HiromuHota: Rename SimpleTokenizer into SimpleParser and let it inherit LingualParser.
- @HiromuHota: Move all ligual parsers into lingual_parser folder.
- @HiromuHota: Make load_lang_model private as a model is internally loaded during init.
- @HiromuHota: Add a unit test for `Parser` with tabular=False. (#261)
- @HiromuHota: Now `longest_match_only` of `Union`, `Intersect`, and `Inverse` override that of child matchers.
- @HiromuHota: Use the official name "beautifulsoup4" instead of an alias "bs4". (#306)
- @HiromuHota: Pin PyTorch on 1.1.0 to align with Snorkel of 0.9.X.
- @HiromuHota: Depend on psycopg2 instead of psycopg2-binary as the latter is not recommended for production.
- @HiromuHota: Change the default value for `delim` of `SimpleParser` from "<NB>" to ".". (#272)

11.7.3 Deprecated

- @HiromuHota: Classifier and its subclass disc_models are deprecated, and in v0.8.0 they will be removed.

11.7.4 Removed

- @HiromuHota: Remove __repr__ from each mixin class as the referenced attributes are not available.
- @HiromuHota: Remove the dependency on nltk, but `PorterStemmer()` can still be used, if it is provided as `DictionaryMatch(stemmer=PorterStemmer())`.
- @HiromuHota: Remove `_NgramMatcher` and `_FigureMatcher` as they are no longer needed.
- @HiromuHota: Remove the dependency on Pandas and visual_linker._display_links.

11.7.5 Fixed

- @senwu: Fix legacy code bug in `SymbolTable`.
- @HiromuHota: Fix the type of max_docs.
- @HiromuHota: Associate sentence with section and paragraph no matter what tabular is. (#261)
- @HiromuHota: Add a safeguard that prevents from accessing Meta.engine before it is assigned. Also this change allows creating a mention/candidate subclass even before Meta is initialized.
- @HiromuHota: Create an Engine and open a connection in each child process. (#323)
- @HiromuHota: Fix `featurizer.apply(docs=train_docs)` fails on clearing. (#250)
- @HiromuHota: Correct abs_char_offsets to make it absolute. (#332)
- @HiromuHota: Fix deadlock error during Labeler.apply and Featurizer.apply. (#328)
- @HiromuHota: Avoid networkx 2.4 so that snorkel-metal does not use the removed API.
- @HiromuHota: Fix the issue that Labeler.apply with docs instead of split fails. (#340)
- @HiromuHota: Make mention/candidate_subclasses and their objects picklable.
- @HiromuHota: Make Visualizer#display_candidates mention-type argnostic.
- @HiromuHota: Ensure labels get updated when LFs are updated. (#336)

11.8 0.7.0 - 2019-06-12

11.8.1 Added

- @HiromuHota: Add notes about the current implementation of data models.
- @HiromuHota: Add Featurizer#upsert_keys.
- @HiromuHota: Update the doc for OS X about an external dependency on libomp.
- @HiromuHota: Add test_classifier.py to unit test Classifier and its subclasses.
- @senwu: Add test_simple_tokenizer.py to unit test simple_tokenizer.
- @HiromuHota: Add test_spacy_parser.py to unit test spacy_parser.

11.8.2 Changed

- @HiromuHota: Assign a section for mention spaces.
- @HiromuHota: Incorporate entity_confusion_matrix as a first-class citizen and rename it to confusion_matrix because it can be used both entity-level and mention-level.
- @HiromuHota: Separate Spacy#_split_sentences_by_char_limit to test itself.
- @HiromuHota: Refactor the custom sentence_boundary_detector for readability and efficiency.
- @HiromuHota: Remove a redundant argument, document, from Spacy#split_sentences.
- @HiromuHota: Refactor TokenPreservingTokenizer for readability.

11.8.3 Removed

- @HiromuHota: Remove `data_model_utils.tabular.same_document`, which always returns True because a candidate can only have mentions from the same document under the current implemention of `CandidateExtractorUDF`.

11.8.4 Fixed

- @senwu: Fix the doc about the PostgreSQL version requirement.

11.9 0.6.2 - 2019-04-01

11.9.1 Fixed

- @lukehsiao: Fix Meta initialization bug which would configure logging upon import rather than allowing the user to configure logging themselves.

11.10 0.6.1 - 2019-03-29

11.10.1 Added

- @senwu: update the spacy version to v2.1.x.
- @lukehsiao: provide `fonduer.init_logging()` as a way to configure logging to a temp directory by default.

Note: Although you can still configure `logging` manually, with this change we also provide a function for initializing logging. For example, you can call:

```
import logging
import fonduer

# Optionally configure logging
fonduer.init_logging(
  log_dir="log_folder",
  format="[%(asctime)s][%(levelname)s] %(name)s:%(lineno)s - %(message)s",
  level=logging.INFO
)

session = fonduer.Meta.init(conn_string).Session()
```

which will create logs within the `log_folder` directory. If logging is not explicitly initialized, we will provide a default configuration which will store logs in a temporary directory.

11.10.2 Changed

- @senwu: Update the whole logging strategy.

Note: For the whole logging strategy:

With this change, the running log is stored `fonduer.log` in the `{fonduer.Meta.log_path}/{datetime}` folder. User can specify it using `fonduer.init_logging()`. It also contains the learning logs init.

For learning logging strategy:

Previously, the model checkpoints are stored in the user provided folder by `save_dir` and the name for checkpoint is `{model_name}.mdl.ckpt.{global_step}`.

With this change, the model is saved in the subfolder of the same folder `fonduer.Meta.log_path` with log file file. Each learning run creates a subfolder under name `{datetime}_{model_name}` with all model checkpoints and tensorboard log file init. To use the tensorboard to check the learning curve, run `tensorboard --logdir LOG_FOLDER`.

11.10.3 Fixed

- @senwu: Change the exception condition to make sure parser run end to end.
- @lukehsiao: Fix parser error when text was located in the `tail` of an LXML table node..
- @HiromuHota: Store lemmas and pos_tags in case they are returned from a tokenizer.
- @HiromuHota: Use unidic instead of ipadic for Japanese. (#231)
- @senwu: Use mecab-python3 version 0.7 for Japanese tokenization since spaCy only support version 0.7.
- @HiromuHota: Use black 18.9b0 or higher to be consistent with isort. (#225)
- @HiromuHota: Workaround no longer required for Japanese as of spaCy v2.1.0. (#224)
- @senwu: Update the metal version.
- @senwu: Expose the `b` and `pos_label` in training.
- @senwu: Fix the issue that pdfinfo causes parsing error when it contains more than one `Page`.

11.11 0.6.0 - 2019-02-17

11.11.1 Changed

- @lukehsiao: improved performance of `data_model_utils` through caching and simplifying the underlying queries. (#212, #215)
- @senwu: upgrade to PyTorch v1.0.0. (#209)

11.11.2 Removed

- @lukehsiao: Removed the redundant `get_gold_labels` function.

Note: Rather than calling get_gold_labels directly, call it from the Labeler:

```
from fonduer.supervision import Labeler
labeler = Labeler(session, [relations])
L_gold_train = labeler.get_gold_labels(train_cands, annotator='gold')
```

Rather than:

```
from fonduer.supervision import Labeler, get_gold_labels
labeler = Labeler(session, [relations])
L_gold_train = get_gold_labels(session, train_cands, annotator_name='gold')
```

11.11.3 Fixed

- @senwu: Improve type checking in featurization.
- @lukehsiao: Fixed sentence.sentence_num bug in get_neighbor_sentence_ngrams.
- @lukehsiao: Add session synchronization to sqlalchemy delete queries. (#214)
- @lukehsiao: Update PyYAML dependency to patch CVE-2017-18342. (#205)
- @KenSugimoto: Fix max/min in `visualizer.get_box`

11.12 0.5.0 - 2019-01-01

11.12.1 Added

- @senwu: Support CSV, TSV, Text input data format. For CSV format, `CSVDocPreprocessor` treats each line in the input file as a document. It assumes that each column is one section and content in each column as one paragraph as default. However, if the column is complex, an advanced parser may be used by specifying `parser_rule` parameter in a dict format where key is the column index and value is the specific parser.

Note:

In Fonduer v0.5.0, you can use `CSVDocPreprocessor`:

```
from fonduer.parser import Parser
from fonduer.parser.preprocessors import CSVDocPreprocessor
from fonduer.utils.utils_parser import column_constructor

max_docs = 10

# Define specific parser for the third column (index 2), which takes␣
↪``text``,
# ``name=None``, ``type="text"``, and ``delim=None`` as input and generate
# ``(content type, content name, content)`` for ``build_node``
# in ``fonduer.utils.utils_parser``.
```

(continues on next page)

(continued from previous page)

```
parser_rule = {
    2: partial(column_constructor, type="figure"),
}

doc_preprocessor = CSVDocPreprocessor(
    PATH_TO_DOCS, max_docs=max_docs, header=True, parser_rule=parser_rule
)

corpus_parser = Parser(session, structural=True, lingual=True, visual=False)
corpus_parser.apply(doc_preprocessor, parallelism=PARALLEL)

all_docs = corpus_parser.get_documents()
```

For TSV format, `TSVDocPreprocessor` assumes each line in input file as a document which should follow (doc_name <tab> doc_text) format.

For Text format, `TextDocPreprocessor` assumes one document per file.

11.12.2 Changed

- @senwu: Reorganize `learning` module to use pytorch dataloader, include `MultiModalDataset` to better handle multimodal information, and simplify the code
- @senwu: Remove `batch_size` input argument from `_calc_logits`, `marginals`, `predict`, and `score` in `Classifier`
- @senwu: Rename `predictions` to `predict` in `Classifier` and update the input arguments to have `pos_label` (assign positive label for binary class prediction) and `return_probs` (If True, return predict probablities as well)
- @senwu: Update `score` function in `Classifier` to include: (1) For binary: precision, recall, F-beta score, accuracy, ROC-AUC score; (2) For categorical: accuracy;
- @senwu: Remove `LabelBalancer`
- @senwu: Remove original `Classifier` class, rename `NoiseAwareModel` to `Classifier` and use the same setting for both binary and multi-class classifier
- @senwu: Unify the loss (`SoftCrossEntropyLoss`) for all settings
- @senwu: Rename `layers` in learning module to `modules`
- @senwu: Update code to use Python 3.6+'s f-strings
- @HiromuHota: Reattach doc with the current session at MentionExtractorUDF#apply to avoid doing so at each MentionSpace.

11.12.3 Fixed

- @HiromuHota: Modify docstring of functions that return get_sparse_matrix
- @lukehsiao: Fix the behavior of `get_last_documents` to return Documents that are correctly linked to the database and can be navigated by the user. (#201)
- @lukehsiao: Fix the behavior of MentionExtractor `clear` and `clear_all` to also delete the Candidates that correspond to the Mentions.

11.13 0.4.1 - 2018-12-12

11.13.1 Added

- @senwu: Added alpha spacy support for Chinese tokenizer.

11.13.2 Changed

- @lukehsiao: Add soft version pinning to avoid failures due to dependency API changes.
- @j-rausch: Change `get_row_ngrams` and `get_col_ngrams` to return `None` if the passed `Mention` argument is not inside a table. (#194)

11.13.3 Fixed

- @senwu: fix non-deterministic issue from get_candidates and get_mentions by parallel candidate/mention generation.

11.14 0.4.0 - 2018-11-27

11.14.1 Added

- @senwu: Rename `span` attribute to `context` in mention_subclass to better support mulitmodal mentions. (#184)

Note: The way to retrieve corresponding data model object from mention changed. In Fonduer v0.3.6, we use `.span`:

```
# sent_mention is a SentenceMention
sentence = sent_mention.span.sentence
```

With this release, we use `.context`:

```
# sent_mention is a SentenceMention
sentence = sent_mention.context.sentence
```

- @senwu: Add support to extract multimodal candidates and add `DoNothingMatcher` matcher. (#184)

Note: The Mention extraction support all data types in data model. In Fonduer v0.3.6, Mention extraction only supports `MentionNgrams` and `MentionFigures`:

```
from fonduer.candidates import (
    MentionFigures,
    MentionNgrams,
)
```

With this release, it supports all data types:

```
from fonduer.candidates import (
    MentionCaptions,
    MentionCells,
    MentionDocuments,
    MentionFigures,
    MentionNgrams,
    MentionParagraphs,
    MentionSections,
    MentionSentences,
    MentionTables,
)
```

- @senwu: Add support to parse multiple sections in parser, fix webpage context, and add name column for each context in data model. (#182)

11.14.2 Fixed

- @senwu: Remove unnecessary backref in mention generation.
- @j-rausch: Improve error handling for invalid row spans. (#183)

11.15 0.3.6 - 2018-11-15

11.15.1 Fixed

- @lukehsiao: Updated snorkel-metal version requirement to ensure new syntax works when a user upgrades Fonduer.
- @lukehsiao: Improve error messages on PostgreSQL connection and update FAQ.

11.16 0.3.5 - 2018-11-04

11.16.1 Added

- @senwu: Add `SparseLSTM` support reducing the memory used by the LSTM for large applications. (#175)

Note: With the SparseLSTM discriminative model, we save memory for the origin LSTM model while sacrificing runtime. In Fonduer v0.3.5, SparseLSTM is as follows:

```
from fonduer.learning import SparseLSTM

disc_model = SparseLSTM()
disc_model.train(
    (train_cands, train_feature), train_marginals, n_epochs=5, lr=0.001
)
```

11.16.2 Fixed

- @senwu: Fix issue with `get_last_documents` returning the incorrect number of docs and update the tests. (#176)
- @senwu: Use the latest MeTaL syntax and fix flake8 issues. (#173)

11.17 0.3.4 - 2018-10-17

11.17.1 Changed

- @senwu: Use `sqlalchemy` to check connection string. Use `postgresql` instead of `postgres` in connection string.

11.17.2 Fixed

- @lukehsiao: The features/labels/gold_label key tables were not properly designed for multiple relations in that they indistinguishably shared the global index of keys. This fixes this issue by including the names of the relations associated with each key. In addition, this ensures that clearing a single relation, or relabeling a single training relation does not inadvertently corrupt the global index of keys. (#167)

11.18 0.3.3 - 2018-09-27

11.18.1 Changed

- @lukehsiao: Added `longest_match_only` parameter to `LambdaFunctionMatcher`, which defaults to False, rather than True. (#165)

11.18.2 Fixed

- @lukehsiao: Fixes the behavior of the `get_between_ngrams` data model util. (#164)
- @lukehsiao: Batch queries so that PostgreSQL buffers aren't exceeded. (#162)

11.19 0.3.2 - 2018-09-20

11.19.1 Changed

- @lukehsiao: `MentionNgrams` `split_tokens` now defaults to an empty list and splits on all occurrences, rather than just the first occurrence.
- @j-rausch: Parser will now skip documents with parsing errors rather than crashing.

11.19.2 Fixed

- @lukehsiao: Fix attribute error when using MentionFigures.

11.20 0.3.1 - 2018-09-18

11.20.1 Fixed

- @lukehsiao: Fix the layers module in fonduer.learning.disc_models.layers.

11.21 0.3.0 - 2018-09-18

11.21.1 Added

- @lukehsiao: Add supporting functions for incremental knowledge base construction. (#154)
- @j-rausch: Added alpha spacy support for Japanese tokenizer.
- @senwu: Add sparse logistic regression support.
- @senwu: Support Python 3.7.
- @lukehsiao: Allow user to change featurization settings by providing `.fonduer-config.yaml` in their project.
- @lukehsiao: Add a new Mention object, and have Candidate objects be composed of Mention objects, rather than directly of Spans. This allows a single Mention to be reused in multiple relations.
- @lukehsiao: Improved connection-string validation for the Meta class.

11.21.2 Changed

- @j-rausch: `Document.text` now returns the modified document text, based on the user-defined html-tag stripping in the parsing stage.
- @j-rausch: `Ngrams` now has a `n_min` argument to specify a minimum number of tokens per extracted n-gram.
- @lukehsiao: Rename `BatchLabelAnnotator` to `Labeler` and `BatchFeatureAnnotator` to `Featurizer`. The classes now support multiple relations.
- @j-rausch: Made spacy tokenizer to default tokenizer, as long as there is (alpha) support for the chosen language. `` `lingual` `` argument now specifies whether additional spacy NLP processing shall be performed.
- @senwu: Reorganize the disc model structure. (#126)

- @lukehsiao: Add `session` and `parallelism` as a parameter to all UDF classes.
- @j-rausch: Sentence splitting in lingual mode is now performed by spacy's sentencizer instead of the dependency parser. This can lead to variations in sentence segmentation and tokenization.
- @j-rausch: Added `language` argument to `Parser` for specification of language used by `spacy_parser`. E.g. `language='en'`.
- @senwu: Change weak supervision learning framework from numbskull to *MeTaL <https://github.com/HazyResearch/metal>_*. (#119)
- @senwu: Change learning framework from Tensorflow to PyTorch. (#115)
- @lukehsiao: Blacklist <script> nodes by default when parsing HTML docs.
- @lukehsiao: Reorganize ReadTheDocs structure to mirror the repository structure. Now, each pipeline phase's user-facing API is clearly shown.
- @lukehsiao: Rather than importing ambiguously from `fonduer` directly, disperse imports into their respective pipeline phases. This eliminates circular dependencies, and makes imports more explicit and clearer to the user where each import is originating from.
- @lukehsiao: Provide debug logging of external subprocess calls.
- @lukehsiao: Use `tdqm` for progress bar (including multiprocessing).
- @lukehsiao: Set the default PostgreSQL client encoding to "utf8".
- @lukehsiao: Organize documentation for `data_model_utils` by modality. (#85)
- @lukehsiao: Rename `lf_helpers` to `data_model_utils`, since they can be applied more generally to throttlers or used for error analysis, and are not limited to just being used in labeling functions.
- @lukehsiao: Update the CHANGELOG to start following KeepAChangelog conventions.

11.21.3 Removed

- @lukehsiao: Remove the XMLMultiDocPreprocessor.
- @lukehsiao: Remove the `reduce` option for UDFs, which were unused.
- @lukehsiao: Remove get parent/children/sentence generator from Context. (#87)
- @lukehsiao: Remove dependency on `pdftotree`, which is currently unused.

11.21.4 Fixed

- @j-rausch: Improve `spacy_parser` performance. We split the lingual parsing pipeline into two stages. First, we parse structure and gather all sentences for a document. Then, we merge and feed all sentences per document into the spacy NLP pipeline for more efficient processing.
- @senwu: Speed-up of `_get_node` using caching.
- @HiromuHota: Fixed bug with Ngram splitting and empty TemporarySpans. (#108, #112)
- @lukehsiao: Fixed PDF path validation when using `visual=True` during parsing.
- @lukehsiao: Fix Meta bug which would not switch databases when init() was called with a new connection string.

Note: With the addition of Mentions, the process of Candidate extraction has changed. In Fonduer v0.2.3, Candidate extraction was as follows:

```
candidate_extractor = CandidateExtractor(PartAttr,
                        [part_ngrams, attr_ngrams],
                        [part_matcher, attr_matcher],
                        candidate_filter=candidate_filter)

candidate_extractor.apply(docs, split=0, parallelism=PARALLEL)
```

With this release, you will now first extract Mentions and then extract Candidates based on those Mentions:

```
# Mention Extraction
part_ngrams = MentionNgramsPart(parts_by_doc=None, n_max=3)
temp_ngrams = MentionNgramsTemp(n_max=2)
volt_ngrams = MentionNgramsVolt(n_max=1)

Part = mention_subclass("Part")
Temp = mention_subclass("Temp")
Volt = mention_subclass("Volt")
mention_extractor = MentionExtractor(
    session,
    [Part, Temp, Volt],
    [part_ngrams, temp_ngrams, volt_ngrams],
    [part_matcher, temp_matcher, volt_matcher],
)
mention_extractor.apply(docs, split=0, parallelism=PARALLEL)

# Candidate Extraction
PartTemp = candidate_subclass("PartTemp", [Part, Temp])
PartVolt = candidate_subclass("PartVolt", [Part, Volt])

candidate_extractor = CandidateExtractor(
    session,
    [PartTemp, PartVolt],
    throttlers=[temp_throttler, volt_throttler]
)

candidate_extractor.apply(docs, split=0, parallelism=PARALLEL)
```

Furthermore, because Candidates are now composed of Mentions rather than directly of Spans, to get the Span object from a mention, use the `.span` attribute of a Mention.

Note: Fonduer has been reorganized to require more explicit import syntax. In Fonduer v0.2.3, nearly everything was imported directly from fonduer:

```
from fonduer import (
    CandidateExtractor,
    DictionaryMatch,
    Document,
    FeatureAnnotator,
    GenerativeModel,
    HTMLDocPreprocessor,
    Intersect,
    LabelAnnotator,
    LambdaFunctionMatcher,
    MentionExtractor,
    Meta,
```

(continues on next page)

(continued from previous page)

```
    Parser,
    RegexMatchSpan,
    Sentence,
    SparseLogisticRegression,
    Union,
    candidate_subclass,
    load_gold_labels,
    mention_subclass,
)
```

With this release, you will now import from each pipeline phase. This makes imports more explicit and allows you to more clearly see which pipeline phase each import is associated with:

```
from fonduer import Meta
from fonduer.candidates import CandidateExtractor, MentionExtractor
from fonduer.candidates.matchers import (
    DictionaryMatch,
    Intersect,
    LambdaFunctionMatcher,
    RegexMatchSpan,
    Union,
)
from fonduer.candidates.models import candidate_subclass, mention_subclass
from fonduer.features import Featurizer
from metal.label_model import LabelModel # GenerativeModel in v0.2.3
from fonduer.learning import SparseLogisticRegression
from fonduer.parser import Parser
from fonduer.parser.models import Document, Sentence
from fonduer.parser.preprocessors import HTMLDocPreprocessor
from fonduer.supervision import Labeler, get_gold_labels
```

11.22 0.2.3 - 2018-07-23

11.22.1 Added

- @lukehsiao: Support Figures nested in Cell contexts and Paragraphs in Figure contexts. (#84)

11.23 0.2.2 - 2018-07-22

Note: Version 0.2.0 and 0.2.1 had to be skipped due to errors in uploading those versions to PyPi. Consequently, v0.2.2 is the version directly after v0.1.8.

Warning: This release is NOT backwards compatable with v0.1.8. The code has now been refactored into submodules, where each submodule corresponds with a phase of the Fonduer pipeline. Consequently, you may need to adjust the paths of your imports from Fonduer.

11.23.1 Added

- @senwu: Add branding, OSX tests. (#61, #62)
- @lukehsiao: Update the Data Model to include Caption, Section, Paragraph. (#76, #77, #78)

11.23.2 Changed

- @senwu: Split up lf_helpers into separate files for each modality. (#81)
- @lukehsiao: Rename to Phrase to Sentence. (#72)
- @lukehsiao: Split models and preprocessors into individual files. (#60, #64)

11.23.3 Removed

- @lukehsiao: Remove the futures imports, truly making Fonduer Python 3 only. Also reorganize the codebase into submodules for each pipeline phase. (#59)

11.23.4 Fixed

- A variety of small bugfixes and code cleanup. (view milestone)

11.24 0.1.8 - 2018-06-01

11.24.1 Added

- @prabh06: Extend styles parsing and add regex search (#52)

11.24.2 Removed

- @senwu: Remove the Viewer, which is unused in Fonduer (#55)
- @lukehsiao: Remove unnecessary encoding in __repr__ (#50)

11.24.3 Fixed

- @senwu: Fix SimpleTokenizer for lingual features are disabled (#53)
- @lukehsiao: Fix LocationMatch NER tags for spaCy (#50)

11.25 0.1.7 - 2018-04-04

Warning: This release is NOT backwards compatable with v0.1.6. Specifically, the `snorkel` submodule in fonduer has been removed. Any previous imports of the form:

```
from fonduer.snorkel._ import _
```

Should drop the `snorkel` submodule:

```
from fonduer._ import _
```

Tip: To leverage the logging output of Fonduer, such as in a Jupyter Notebook, you can configure a logger in your application:

```
import logging

logging.basicConfig(stream=sys.stdout, format='[%(levelname)s] %(name)s - %(message)s
↪')
log = logging.getLogger('fonduer')
log.setLevel(logging.INFO)
```

11.25.1 Added

- @lukehsiao: Add lf_helpers to ReadTheDocs (#42)

11.25.2 Removed

- @lukehsiao: Remove SQLite code, switch to logging, and absorb snorkel codebase directly into the fonduer package for simplicity (#44)
- @lukehsiao: Remove unused package dependencies (#41)

11.26 0.1.6 - 2018-03-31

11.26.1 Changed

- @lukehsiao: Switch README from Markdown to reStructuredText

11.26.2 Fixed

- @senwu: Fix support for providing a PostgreSQL username and password as part of the connection string provided to Meta.init() (#40)

11.27 0.1.5 - 2018-03-31

Warning: This release is NOT backwards compatable with v0.1.4. Specifically, in order to initialize a session with postgresql, you no longer do

```
os.environ['SNORKELDB'] = 'postgres://localhost:5432/' + DBNAME
from fonduer import SnorkelSession
session = SnorkelSession()
```

which had the side-effects of manipulating your database tables on import (or creating a `snorkel.db` file if you forgot to set the environment variable). Now, you use the Meta class to initialize your session:

```
from fonduer import Meta
session = Meta.init("postgres://localhost:5432/" + DBNAME).Session()
```

No side-effects occur until `Meta` is initialized.

11.27.1 Removed

- @lukehsiao: Remove reliance on environment vars and remove side-effects of importing fonduer (#36)

11.27.2 Fixed

- @lukehsiao: Bring codebase in PEP8 compliance and add automatic code-style checks (#37)

11.28 0.1.4 - 2018-03-30

11.28.1 Changed

- @lukehsiao: Separate tutorials into their own repo (#31)

11.29 0.1.3 - 2018-03-29

11.29.1 Fixed

Minor hotfix to the README formatting for PyPi.

11.30 0.1.2 - 2018-03-29

11.30.1 Added

- @lukehsiao: Deploy Fonduer to PyPi using Travis-CI

CHAPTER

TWELVE

INSTALLATION

To test changes in the package, you install it in editable mode locally in your virtualenv by running:

```
$ make dev
```

This will also install our pre-commit hooks and local packages needed for style checks.

Tip: If you need to install a locally edited version of fonduer in a separate location, such as an application, you can directly install your locally modified version:

```
$ pip install -e path/to/fonduer/
```

in the virtualenv of your application.

CHAPTER

THIRTEEN

TESTING

We use pytest to run our tests. Our tests are all located in the `tests` directory in the repo, and are meant to be run *after* installing Fonduer locally.

You need PostgreSQL running locally with trust authentication enabled. You'll also need to download the external test data:

```
$ cd tests/
$ ./download_data.sh
$ cd ..
```

Then, you'll be able to run our tests:

```
$ make test
```

CHAPTER

FOURTEEN

CODE STYLE

For code consistency, we have a pre-commit configuration file so that you can easily install pre-commit hooks to run style checks before you commit your files. You can setup our pre-commit hooks by running:

```
$ pip install -r requirements-dev.txt
$ pre-commit install
```

Or, just run:

```
$ make dev
```

Now, each time you commit, checks will be run using the packages explained below.

We use black as our Python code formatter with its default settings. Black helps minimize the line diffs and allows you to not worry about formatting during your own development. Just run black on each of your files before committing them.

Tip: Whatever editor you use, we recommend checking out black editor integrations to help make the code formatting process just a few keystrokes.

For sorting imports, we reply on isort. Our repository already includes a *.isort.cfg* that is compatible with black. You can run a code style check on your local machine by running our checks:

```
$ make check
```

14.1 Docstring format

We use Sphinx to build documentation. While Sphinx's `autodoc` extension, which automatically generates documentation from docstring, supports several docstring formats, we use Sphinx docstring format. A typical Sphinx docstring looks like the following:

Listing 1: A typical Sphinx docstring

```
"""[Summary]

:param [ParamName]: [ParamDescription], defaults to [DefaultParamVal]
:type [ParamName]: [ParamType](, optional)
...
:raises [ErrorType]: [ErrorDescription]
...
```

(continues on next page)

(continued from previous page)

```
:return: [ReturnDescription]
:rtype: [ReturnType]
"""
```

where *:type* and *:rtype* can be omitted. Since we annotate functions with types (PEP 484), *:type* and *:rtype* in the docstring are redundant and hence not allowed in Fonduer. An example Fonduer docstring looks like:

Listing 2: An example Fonduer docstring

```
def greeting(name: str) -> str:
    """Return a greeting to a person.

    :param name: a person's name to greet.
    :return: a greeting string.
    """
    return 'Hello ' + name
```

Note that *:type* and *:rtype* are omitted and that the function is annotated with types.

CHAPTER

FIFTEEN

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of DARPA under No. N66001-15-C-4043 (SIMPLEX), No. FA8750-17-2-0095 (D3M), No. FA8750-12-2-0335, and No. FA8750-13-2-0039; DOE under 108845; NIH under U54EB020405; ONR under No. N000141712266 and No. N000141310129; the Intel/NSF CPS Security grant No. 1505728; the Secure Internet of Things Project; Qualcomm; Ericsson; Analog Devices; the National Science Foundation Graduate Research Fellowship under Grant No. DGE-114747; the Stanford Finch Family Fellowship; the Moore Foundation; the Okawa Research Grant; American Family Insurance; Accenture; Toshiba; and members of the Stanford DAWN project: Intel, Microsoft, Google, Teradata, and VMware. We thank Holly Chiang, Bryan He, and Yuhao Zhang for helpful discussions. We also thank Prabal Dutta, Mark Horowitz, and Björn Hartmann for their feedback on early versions of this work. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, DOE, NIH, ONR, or the U.S. Government.

PYTHON MODULE INDEX

f

INDEX

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X