What's prodigy?

prodigy is a modern annotation tool for creating training data for machine learning models. It runs entirely on the user's own machine and is **fully scriptable** in Python.

prodigy includes a CLI and Python library that lets you run an annotation server, and serve a web app with a collection of interfaces to create annotations for text, HTML, images and audio data.

prodigy includes various pre-built workflows, called **recipes**, that load the input data, use a given interface to create the annotations and implement various logic for automation.

```
recipe.py                                                    PSEUDOCODE

import prodigy

@prodigy.recipe(                                  ← Prodigy's decorator
    "my-custom-recipe",
    dataset=("Dataset to save answers to", "positional", None, str),
    view_id=("Annotation interface", "option", "v", str)
)
def my_custom_recipe(dataset, view_id="text"):
    # Load your own streams from anywhere you want
    stream = load_my_custom_stream()              ← Load stream

    def update(examples):
        # This function is triggered when Prodigy receives annotations
        print(f"Received {len(examples)} annotations!")

    return {                                      ← Return dict of components
        "dataset": dataset,
        "view_id": view_id,
        "stream": stream,
        "update": update
    }
```

Update callback, can update model-in-the-loop

Custom recipes let you integrate machine learning models using any framework of your choice, load in data from different sources, implement your own storage solution or add other hooks and features. No matter how complex your pipeline is – if you can call it from a Python function, you can use it in Prodigy.

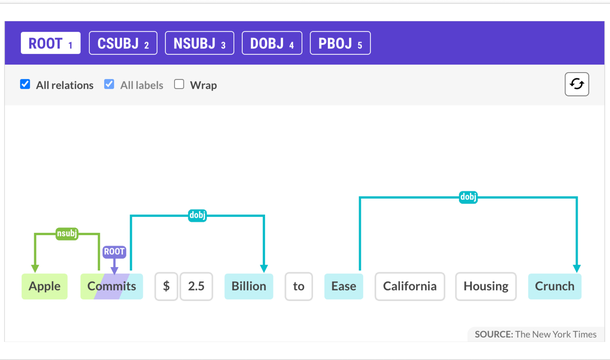
How to use prodigy?

Start the annotation server

```
$ prodigy  dep.correct  deps_news  en_core_web_sm  ./news_headlines.jsonl
  --label ROOT,csubj,nsubj,dobj,pboj  --update
```

Recipe used · spaCy model · Source data · Update model

Label examples



Data-to-spaCy and spaCy Train

```
$ prodigy  data-to-spacy  ./corpus
  --ner news_ner_person,news_ner_org,news_ner_product
  --textcat news_cats2018,news_cats2019  --eval-split 0.3

ℹ Using language 'en'

============================== Generating data ==============================
Components: ner, textcat
Merging training and evaluation data for 2 components
  - [ner] Training: 685 | Evaluation: 300 (from datasets)
  - [textcat] Training: 538 | Evaluation: 230 (30% split)
Training: 1223 | Evaluation: 530
Labels: ner (3) | textcat (5)
✔ Saved 1223 training examples
./corpus/train.spacy
```

prodigy integrates with our NLP library **spaCy** out-of-the-box and provides workflows for creating training data for spaCy models. But since it's extensible, you can develop your own workflows with another framework or export out your data.

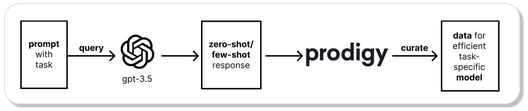
```
============================== Finalizing export ==============================
✔ Saved training config
./corpus/config.cfg

To use this data for training with spaCy, you can run:
python -m spacy train ./corpus/config.cfg --paths.train
./corpus/train.spacy --paths.dev ./corpus/dev.spacy
```

What recipes are built-in?

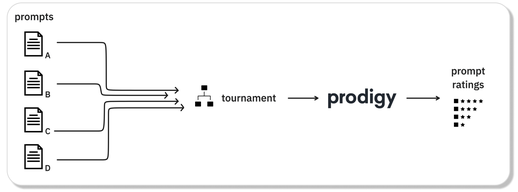
- Named Entity Recognition
- Span Categorization
- Text Classification
- Dependencies & Relations
- Computer Vision
- Audio & Video
- A/B Evaluation

What's new in v1.12?



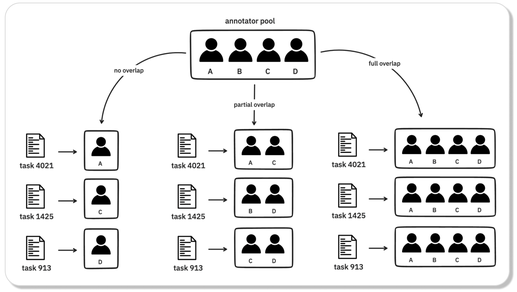
LLM & Prompt Engineering

prodigy supports recipes and tools for Large Language Models and prompt engineering via prompt tournaments. In these tournaments, your prompts will compete as you annotate the one that gives the best results.

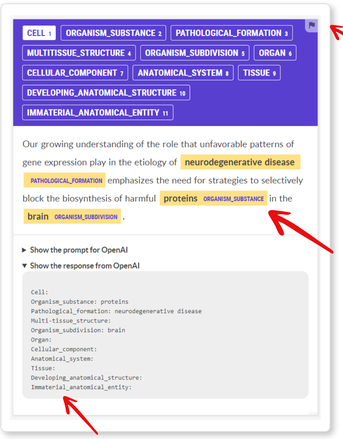


Task Routing

prodigy allows you to distribute the annotation workload among multiple annotators or workers. You can make sure examples are seen by multiple annotators or even set up **custom routing** so that specific examples get seen by specific annotators.



Prodigy's OpenAI NER Correct recipe



Flag annotations as exemplars for few-shot prompting

Suggestions from LLM

Raw response from LLM

Scan here to watch Vincent's v1.12 video

```
Task router based on model confidence                        PSEUDOCODE

def task_router_conf(ctrl: Controller, session_id: str, item: Dict) -> List[str]:
    """Route tasks based on the confidence of a model."""
    # Get all sessions known to the Controller now
    all_annotators = ctrl.all_session_ids

    # Calculate a confidence score from a custom model
    confidence_score = model(item['text'])

    # If the confidence is low, the example might be hard
    # and then everyone needs to check
    if confidence_score < 0.3:                    ← Assign based on model confidence
        return all_annotators

    # Otherwise just one person needs to check.
    # We re-use the task_hash to ensure consistent routing of the task.
    idx = item['_task_hash'] % len(all_annotators)
    # Return list with a single annotator reference
    return [all_annotators[idx]]
```

Example of custom task router

What's future work?

```
[nlp]
lang = "en"
pipeline = ["llm"]

[components]

[components.llm]
factory = "llm"

[components.llm.task]                             ← Add a text classifier using GPT-3.5
@llm_tasks = "spacy.TextCat.v1"
labels = COMPLIMENT,INSULT

[components.llm.backend]
@llm_backends = "spacy.REST.v1"
api = "OpenAI"
config = {"model": "text-davinci-003", "temperature": 0.3}
```

spaCy-LLM config file

The team is working hard on several large projects. For spaCy, we just released **spaCy v3.6** and **spacy-llm v0.4**, an extension for integrating LLMs into structured NLP pipelines. For Prodigy, we're working on **Prodigy v2.0**, which includes enhanced data validation, spaCy-llm integration, built-in IAA metrics, and Structured examples, and much more. We're also restarting Beta testing for **Prodigy Teams**, our long-awaited SaaS version of Prodigy. We're excited for a public release soon!

Scan here to request a free Prodigy research license