Potato

Release 0.1

Blablablab

Feb 17, 2023

CONTENTS

Potato is a **fully free** data annotation tool supporting a wide range of features throughout your entire annotation pipeline.

Note: This project is under active development.

CHAPTER ONE

CONTENTS

1.1 Quick start

Clone the github repo to your computer

```
git clone https://github.com/davidjurgens/potato.git
```

Install all the required dependencies

```
pip3 install -r requirements.txt
```

To run a simple check-box style annotation on text data, run

```
python3 potato/flask_server.py config/examples/simple-check-box.yaml -p 8000
```

This will launch the webserver on port 8000 which can be accessed at http://localhost:8000.

1.2 Example projects (project hub)

Potato aims to improve the replicability of data annotation and reduce the cost for researchers to set up new annotation tasks. Therefore, Potato comes with a list of predefined example projects, and welcome public contribution to the project hub. If you have used potato for your own annotation, you are encouraged to create a pull request and release your annotation setup.

1.2.1 Dialogue analysis (span + categorization)

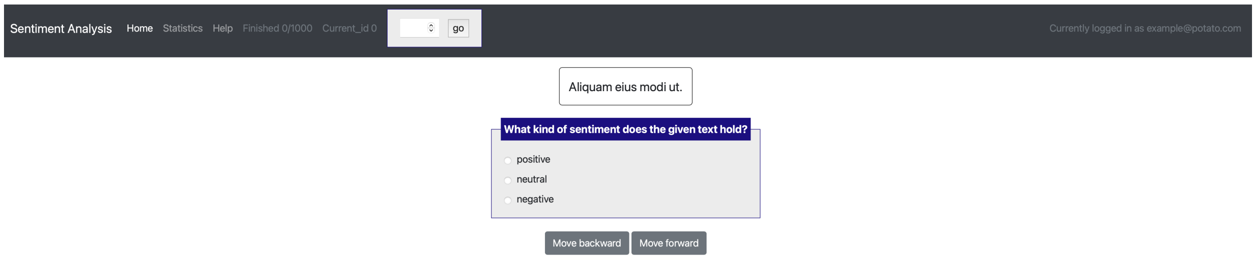
yaml config

```
[launch] python3 potato/flask_server.py example-projects/dialogue_analysis/configs/
↪dialogue-analysis.yaml -p 8000
[Annotate] http://localhost:8000
```

1.2.2 Sentiment analysis (categorization)

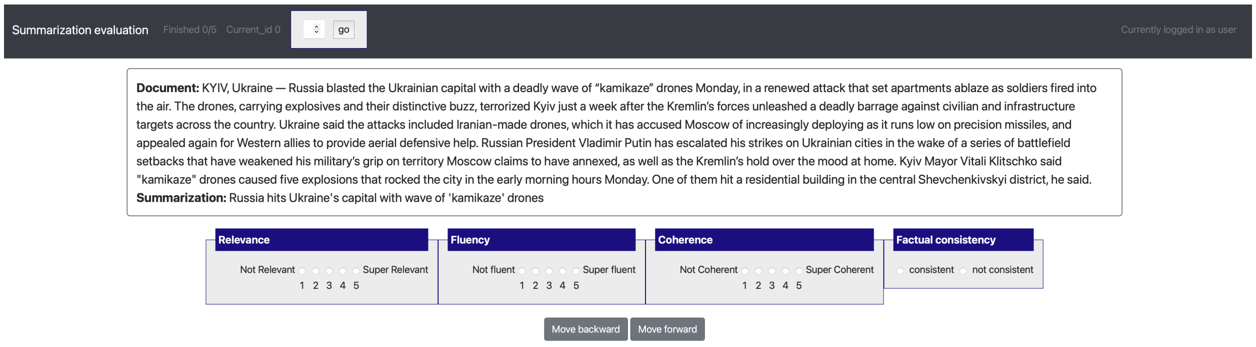
yaml config

```
[launch] python3 potato/flask_server.py example-projects/sentiment_analysis/configs/
↪sentiment-analysis.yaml -p 8000
[Annotate] http://localhost:8000
```



1.2.3 Summarization evaluation (likert + categorization)

```
[launch] python3 potato/flask_server.py example-projects/summarization_evaluation/
↪configs/summ-eval.yaml -p 8000
[Annotate] http://localhost:8000/?PROLIFIC_PID=user
```



Document: KYIV, Ukraine — Russia blasted the Ukrainian capital with a deadly wave of "kamikaze" drones Monday, in a renewed attack that set apartments ablaze as soldiers fired into the air. The drones, carrying explosives and their distinctive buzz, terrorized Kyiv just a week after the Kremlin's forces unleashed a deadly barrage against civilian and infrastructure targets across the country. Ukraine said the attacks included Iranian-made drones, which it has accused Moscow of increasingly deploying as it runs low on precision missiles, and appealed again for Western allies to provide aerial defensive help. Russian President Vladimir Putin has escalated his strikes on Ukrainian cities in the wake of a series of battlefield setbacks that have weakened his military's grip on territory Moscow claims to have annexed, as well as the Kremlin's hold over the mood at home. Kyiv Mayor Vitali Klitschko said "kamikaze" drones caused five explosions that rocked the city in the early morning hours Monday. One of them hit a residential building in the central Shevchenkivskyi district, he said.
Summarization: Russia hits Ukraine's capital with wave of 'kamikaze' drones

Relevance
Not Relevant 1 2 3 4 5 Super Relevant
Fluency
Not fluent 1 2 3 4 5 Super fluent
Coherence
Not Coherent 1 2 3 4 5 Super Coherent
Factual consistency
consistent
not consistent
Move backward
Move forward

1.2.4 Match findings in papers and news (likert + prescreening questions + multi-task)

yaml config | Paper | Dataset

```
[Setup configuration files for multiple similar tasks] python3 potato/setup_multitask_
↪config.py example-projects/match_finding/multitask_config.yaml
[launch] python3 potato/flask_server.py example-projects/match_finding/configs/Computer_
↪Science.yaml -p 8000
[Annotate] http://localhost:8000/?PROLIFIC_PID=user
```

1.2.5 Match findings in papers and news (prestudy test)

yaml config

```
[launch] python3 potato/flask_server.py example-projects/match_finding_with_prestudy/
↪configs/match_finding.yaml -p 8000
[Annotate] http://localhost:8000/?PROLIFIC_PID=user
```

1.2.6 Textual uncertainty (likert + categorization)

yaml config | Paper | Dataset

```
[launch sentence-level] python3 potato/flask_server.py example-projects/textual_
↪uncertainty/configs/sentence_level.yaml -p 8000
[launch aspect-level] python3 potato/flask_server.py example-projects/textual_
↪uncertainty/configs/aspect_level.yaml -p 8000
[Annotate] http://localhost:8000
```

1.2.7 Immigration framing in tweets (Multi-schema categorization)

yaml config | Paper | Dataset

```
[launch] python3 potato/flask_server.py example-projects/immigration_framing/configs/
↪config.yaml -p 8000
[Annotate] http://localhost:8000/
```

1.2.8 GIF Reply Appropriateness (video as label)

yaml config | Paper | Dataset

```
[launch] python3 potato/flask_server.py example-projects/gif_reply/configs/gif-reply.
↪yaml -p 8000
[Annotate] http://localhost:8000/
```

1.3 Usage

Getting started with Potato is easy! Here's what you need to do:

1.3.1 Install Potato to your machine

Potato has a Python-based server architecture that can be run locally or hosted on any device. In order to install Potato:

- Make sure you have Python version 3 installed
- Follow the quickstart instructions here.

1.3.2 Set up the project data

In order to input document and specify:

- Prepare your input data (csv, tsv, or json) and upload it in the `data` folder
- Specify where and in what format you want the output data
- Optional: Update the config YAML file with input and output data preferences

1.3.3 Create your codebook and schema

Next, you'll need to specify what annotators annotate:

- Create your annotation codebook and link it to the annotation interface
- Specify the schema, including:
  - Annotation Type: `multiselect` (checkboxes), `radio` (single selection), `likert` (scale with endpoints labeled), or `text` (free-form)
  - Questions for annotators
  - Answer Choices for multiselect and radio types
  - End Labels and Length for likert type questions
  - Optional Question Features: `required`, `horizontal` (placement of answers is horizontal not vertical), `has_free_response` (whether to include an open text box at the end of multiselect or radio question, like having an "other" option)
  - Optional Answer Features: tooltips, keyboard shortcuts, keywords to highlight
- Optional: Format the schema for the YAML config file (basic examples, advanced examples)

1.3.4 Define annotation settings

There are a few other settings you can play with:

- Choose an existing html template for the annotation task or create a new one
- Optional: Specify privacy and access settings for the task
- Optional: Update the YAML file with the look and feel.
- Optional: Set up active learning

1.3.5 Launch potato locally

And that's it! You can go ahead and get started labeling data in one of two ways:

Option 1: Follow the prompts given above to define a YAML file that specifies the data sources, server configuration, annotation schemes, and any custom visualizations (examples) and then launch potato.

```
python3 potato/flask_server.py config/examples/simple-check-box.yaml -p 8000
```

Option 2: Launch potato without a YAML. In this case, the server will have you follow a series of prompts about the task and automatically generate a YAML file for you. A YAML file is then passed to the server on the command line to launch the server for annotation.

```
python3 potato/flask_server.py -p 8000
```

This will launch the webserver on port 8000 which can be accessed at http://localhost:8000. You can create an account and start labeling data. Clicking "Submit" will autoadvance to the next instance and you can navigate between items using the arrow keys. Potato currently supports one annotation task per server instance, though multiple servers may be run on different posts to concurrently annotate different data.

1.4 Data Formats

1.4.1 Prepare your input data

Upload one or more files containing documents to be annotated in the `data` folder.

We support multiple formats of raw data files, including: csv, tsv, json, or jsonl.

Each document needs, at minimum, a unique identifier and the body of the document.

You can find example data files here. We currently support four different document formats:

- Text: body is the document plaintext (example)
- Image, Video, or GIF: body is the filepath (example)
- Dialogue or a list of text: body is a list of comma-seperated documents and potato will automatically display the list of text horizontally. (example)
- Pairs of text displayed in separate boxes: body is a dictionary of documents (example)
- Best-Worst Scaling: body is a comma-separated list of documents to order (example)
- Custom Arguments: body is one of the above + extra fields for whatever custom arguments you want to enter (example – in this `kwargs` and `other_kwargs` are the custom endpoints for a Likert scale)
- Annotating Document A in context of Document B: body is document A + extra `context` field with the body of document B (example)

You can also use html tags to design the way your text to be displayed. In the match finding example project, html tags are used to create two seperate boxes for the finding pairs.

1.4.2 Update input data formats on the YAML config file

You would pass the input data paths and field names into the YAML config file as follows:

```
# Pass in a comma-separated list of data files containing documents to be annotated in␣
↪this task
"data_files": [
   "data/toy-example1.json",
   "data/toy-example2.json"
],

# Specify the field names containing the document unique identifier (id) and document␣
↪body (text)
"item_properties": {
    "id_key": "id",
    "text_key": "text"
},
```

1.4.3 Update output data preferences on the YAML config file

The output file will include each labeled document's id and annotations; the header will consist of the question and answer labels specified in the schema. You need to specify a subdirectory of the `annotation_output` directory where files for each annotator should be placed. We support multiple output formats, including: csv, tsv, json, or jsonl.

```
# Potato will write the annotation file for all annotations to this
# directory, as well as per-annotator output files and state information
# necessary to restart annotation.
"output_annotation_dir": "annotation_output/folder_name/",

# The output format for the all-annotator data. Allowed formats are:
# * jsonl
# * json (same output as jsonl)
# * csv
# * tsv
#
"output_annotation_format": "json",
```

1.5 Templates and Schemas

`potato` allows deployers to select one or more forms of annotation for their data using predefined schema types in the `"annotation_schemes"` field of the config yaml.

Deployers fill out which options should be shown and then each scheme is rendered into HTML upon the completion of loading data. These schema configurations allow deployers to quickly add keyboard shortcuts to specific options or tooltips to help annotators.

1.5.1 Existing Task templates

Templates for some existing tasks are available:

- Question Answering: yaml config, data example
- Sentiment Analysis: yaml config, data example
- Animated GIF Appropriateness Annotation: yaml config, data example
- Single (Radio) Choice with Active Learning: yaml config, data example

1.5.2 Supported Schemas

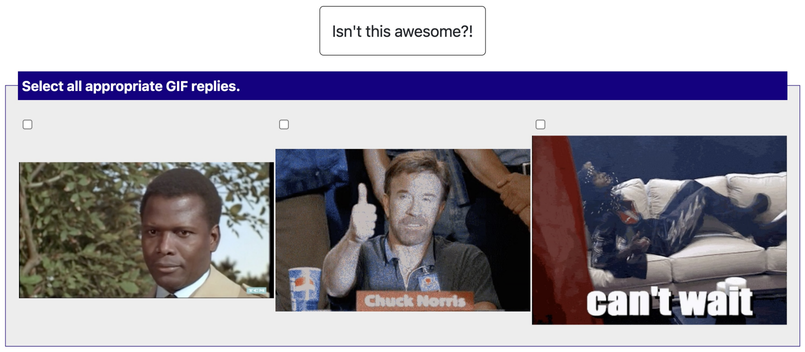
potato currently support 4 customizable schemas with examples are shown below.

Multiple Choice

Simple Checkbox Example (yaml config, data example):

```
"annotation_schemes": [
        {
            "annotation_type": "multiselect",
            "name": "favorite_color",
            "description": "What colors are mentioned in the text?",
            "labels": [
               "blue", "maize", "green", "white"
            ],

            # If true, numbers [1-len(labels)] will be bound to each
            # label. Check box annotations with more than 10 are not supported
            # with this simple keybinding and will need to use the full item
            # specification to bind all labels to keys.
            "sequential_key_binding": True,
        },
]
```

Video as label We also support using video/animated-gif as label for multi-modal annotation (yaml config, data example):



```
"annotation_schemes": [
      {
          "annotation_type": "multiselect",
          "name": "GIF Reply Appropriateness",
          "video_as_label": "True", # <- set this to True for video_as_label annotation
          "description": "Select all appropriate GIF replies.",

          # Files http://[server]:[port]/data/* will be forwarded from directory data/
↪files/*
          "labels": [
             {"name": "{{instance_obj.gifs[0]}}", "videopath": "/files/{{instance_obj.
↪gifs_path[0]}}"},
             {"name": "{{instance_obj.gifs[1]}}", "videopath": "/files/{{instance_obj.
↪gifs_path[1]}}"},
             {"name": "{{instance_obj.gifs[2]}}", "videopath": "/files/{{instance_obj.
↪gifs_path[2]}}"},
          ],

          # If true, numbers [1-len(labels)] will be bound to each
          # label. Check box annotations with more than 10 are not supported
          # with this simple keybinding and will need to use the full item
          # specification to bind all labels to keys.
          "sequential_key_binding": True,
      },
  ],
```

Multiple Choice with Free Response (yaml config, data example):

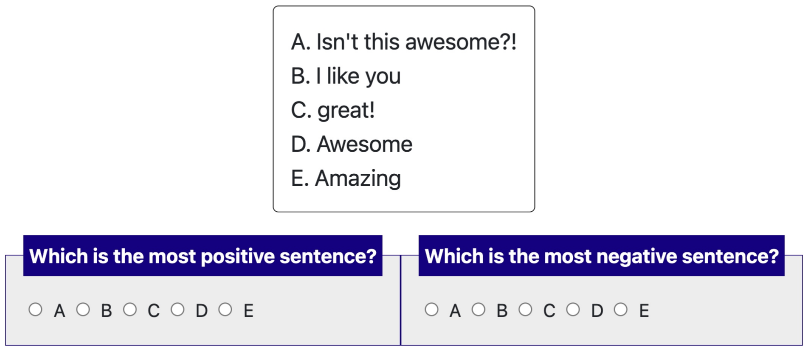
```
"annotation_schemes": [
      {
          "annotation_type": "multiselect",
          "name": "favorite_color",
          "description": "What colors are mentioned in the text?",
          "labels": [
             "blue", "maize", "green", "white"
          ],

          # If true, the field will have an optional text box the user can
          'has_free_response': True,

          # If true, numbers [1-len(labels)] will be bound to each
          # label. Check box annotations with more than 10 are not supported
          # with this simple keybinding and will need to use the full item
          # specification to bind all labels to keys.
          "sequential_key_binding": True,
      },
  ],
```

Single Choice (Radio)

Simple Single (radio) Choice Example (yaml config, data example):

```
"annotation_schemes": [
      {
          "annotation_type": "radio",
          "name": "favorite_color",
          "description": "What food does this text make you want to eat?",
          "labels": [
             "pizza", "bagels", "burgers", "curry", "tacos",
          ],
          # If true, numbers [1-len(labels)] will be bound to each
          # label. Check box annotations with more than 10 are not supported
          # with this simple keybinding and will need to use the full item
          # specification to bind all labels to keys.
          "sequential_key_binding": True,
      },
  ]
```

Best-Worst Scaling Example (yaml config, data example):


A. Isn't this awesome?!
B. I like you
C. great!
D. Awesome
E. Amazing

Which is the most positive sentence?

○ A ○ B ○ C ○ D ○ E

Which is the most negative sentence?

○ A ○ B ○ C ○ D ○ E

```
"annotation_schemes": [
      {
          "annotation_type": "radio",
          "name": "bws_best",
          "description": "Which is the most positive sentence?",

          # If true, display the labels horizontally
          "horizontal": True,

          "labels": [
             "A", "B", "C", "D", "E",
          ],
          "sequential_key_binding": True,
      },

      {
        "annotation_type": "radio",
        "name": "bws_worst",
        "description": "Which is the most negative sentence?",
```

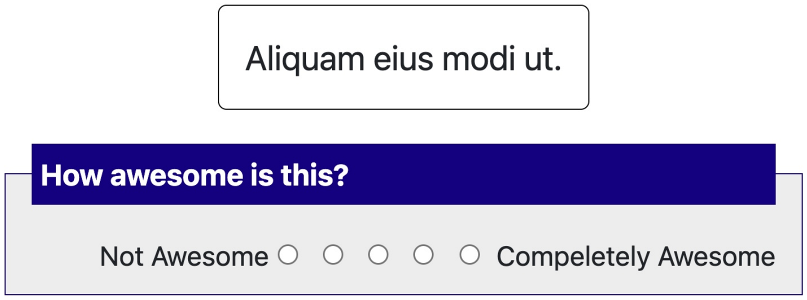
(continues on next page)

(continued from previous page)

```
        # If true, display the labels horizontally
        "horizontal": True,

        "labels": [
          "A", "B", "C", "D", "E",
        ],
        "sequential_key_binding": True,
      },
  ]
```

Likert

Simple Likert Example (yaml config, data example):


Aliquam eius modi ut.

How awesome is this?

Not Awesome ○ ○ ○ ○ ○ Compeletely Awesome

```
"annotation_schemes": [
      {
          "annotation_type": "likert",

          # This name gets used in reporting the annotation results
          "name": "awesomeness",

          # This text is shown to the user and can be a longer statement
          "description": "How awesome is this?",

          # The min and max labels are text shown at each end of the scale
          "min_label": "Not Awesome",
          "max_label": "Compeletely Awesome",

          # How many scale points to show
          "size": 5,

          # If true, keys [1-size] will be bound to scale responses. Likert
          # scales larger than 10 are not supported with this simple
          # keybinding and will need to use the full item specification to
          # bind all scale points to keys.
          "sequential_key_binding": True,
      }
  ]
```

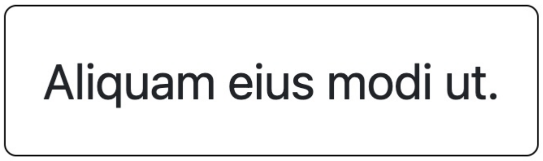
Text span

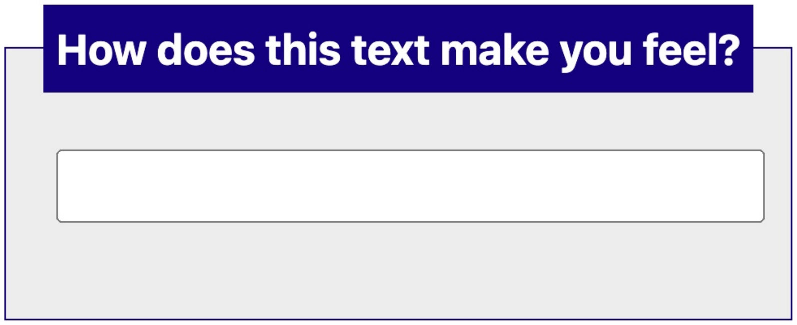
Simple Text Span Example (yaml config, data example):

```
"annotation_schemes": [
    {
        "annotation_type": "highlight",
        "name": "certainty",
        "description": "Highlight which phrases make the sentence more or less certain",
        "labels": [
           "certain", "uncertain"
        ],

        # If true, numbers [1-len(labels)] will be bound to each
        # label. Highlight selection annotations with more than 10 are not supported
        # with this simple keybinding and will need to use the full item
        # specification to bind all labels to keys.
        "sequential_key_binding": True,
    },
],
```

Text Box

Simple Text Box Example (yaml config, data example):





```
"annotation_schemes": [
    {
        "annotation_type": "text",
        "name": "textbox_input",
        "description": "How does this text make you feel?",
    }
]
```

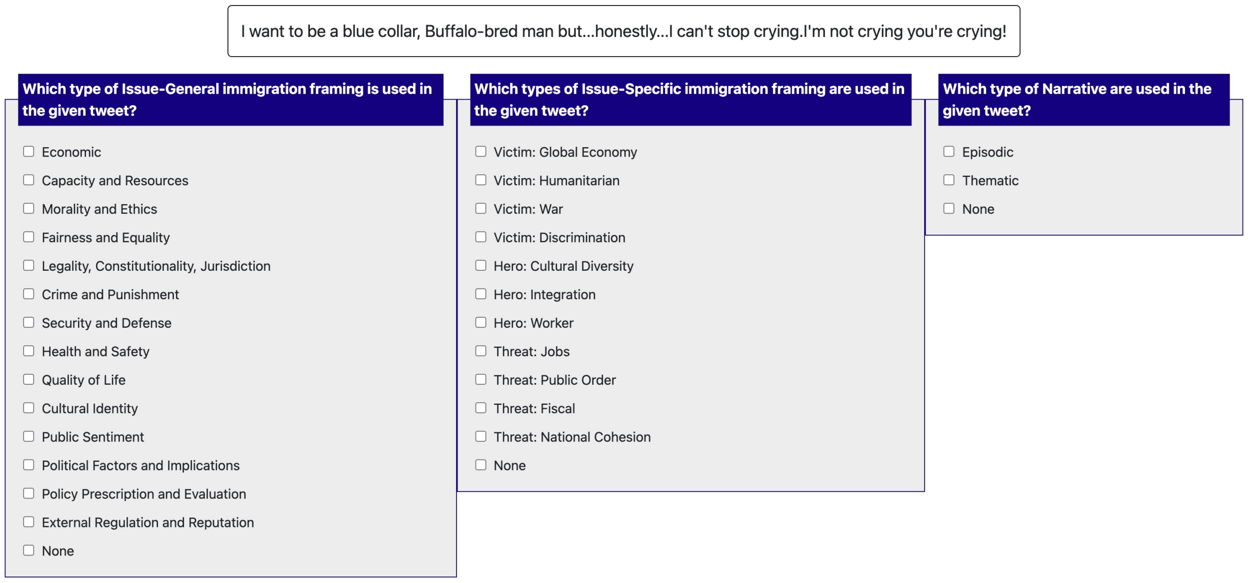
Pairwise comparison

Simple Pairwise Example (yaml config, data example):

```
#setting up list_as_text input, where the pairs will be inputed as a list of texts and␣
↪potato will automatically unfold them
"list_as_text": {
  "text_list_prefix_type": 'alphabet'
},

"annotation_schemes": [
    {
        "annotation_type": "text",
        "name": "textbox_input",
        "description": "How does this text make you feel?",
    }
]
```

1.5.3 Tasks with multiple schemas

potato also support using multiple (different) schemas per annotation task as shown below:


I want to be a blue collar, Buffalo-bred man but...honestly...I can't stop crying.I'm not crying you're crying!

Which type of Issue-General immigration framing is used in the given tweet?

- ☐ Economic
- ☐ Capacity and Resources
- ☐ Morality and Ethics
- ☐ Fairness and Equality
- ☐ Legality, Constitutionality, Jurisdiction
- ☐ Crime and Punishment
- ☐ Security and Defense
- ☐ Health and Safety
- ☐ Quality of Life
- ☐ Cultural Identity
- ☐ Public Sentiment
- ☐ Political Factors and Implications
- ☐ Policy Prescription and Evaluation
- ☐ External Regulation and Reputation
- ☐ None

Which types of Issue-Specific immigration framing are used in the given tweet?

- ☐ Victim: Global Economy
- ☐ Victim: Humanitarian
- ☐ Victim: War
- ☐ Victim: Discrimination
- ☐ Hero: Cultural Diversity
- ☐ Hero: Integration
- ☐ Hero: Worker
- ☐ Threat: Jobs
- ☐ Threat: Public Order
- ☐ Threat: Fiscal
- ☐ Threat: National Cohesion
- ☐ None

Which type of Narrative are used in the given tweet?

- ☐ Episodic
- ☐ Thematic
- ☐ None

```
"annotation_schemes": [
      {
          "annotation_type": "multiselect",
          "single_select":"True",
          "name": "Issue-General",
          "labels": [
              {
                "name": "Economic",
                "tooltip_file": "config/tooltips/ig_economic.html",
                "key_value": '1'
```

(continues on next page)

(continued from previous page)

```
                },
                # ...
            ]
        },
        {
            "annotation_type": "multiselect",
            "name": "Issue-Specific",
            "labels": [

                {
                  "name": "Victim: Global Economy",
                  "tooltip_file": "config/tooltips/sp_global.html"
                },
                # ...
            ]

        },
        # ... more schemes
    ],
```

1.5.4 Add the codebook to the page

If you have a url to a codebook (e.g., in Google Docs), you can add it to the page by setting the `annotation_codebook_url` field in the YAML file. You can also add the task name as the page title using the `annotation_task_name` field.

```
# page title
"annotation_task_name": "Example Task",

# If annotators are using a codebook, this will be linked at the top to the
# instance for easy access
"annotation_codebook_url": "https://www.codebook.com",
```

1.5.5 Choose (or create) your HTML template

Set up the annotation interface by picking an existing HTML template (examples) or creating a custom template:

- `templates/examples/plain_layout.html`: this template covers a wide range of NLP tasks (e.g., text classification, image or gif classification, Likert scales, best-worst scaling, question answering, multiple questions), and is designed to minimize scrolling and optimize placement of the document and questions on the screen.
- `templates/quotes.html`: this template specifies the layout when you want to annotate, not a standalone document, but a document in context of some other document (e.g., if you're annotating replies to a post, and want to show the original post)
- `templates/examples/kwargs_example.html`: this template specifies the layout for a task where each document is rated on some Likert scales with differing endpoints (`kwargs` and `other_kwargs`). It's an example of how to use a custom keyword argument in an HTML file.
- Custom: Create an HTML file that lays out your task pieces and upload it to `potato/templates/`. The templates can be easily customized using JINJA expressions to specify where parts of the annotation task and data are populated within the user-defined template. (custom example 1, custom example 2)

1.5.6 Update YAML file with look and feel

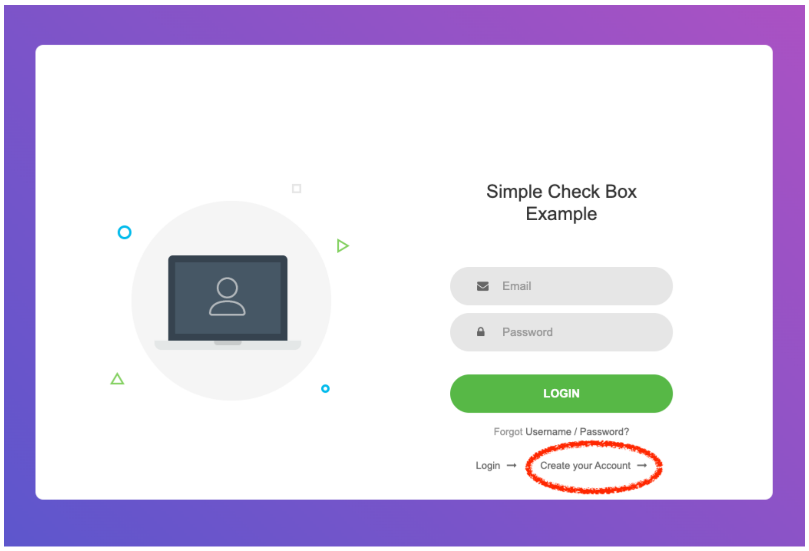
In the YAML file, you'll need to specify what the annotation interface looks like. The `html_layout` field can be updated per the prior section. The rest of the fields can generally be left untouched.

```
# The html that changes the visualiztation for your task. Change this file
# to influence the layout and description of your task. This is not a full
# HTML page, just the piece that does lays out your task's pieces
"html_layout": "templates/examples/plain_layout.html",

# The core UI files for Potato. You should not need to change these normally.
#
# Exceptions to this might include:
# 1) You want to add custom CSS/fonts to style your task
# 2) Your layout requires additional JS/assets to render
# 3) You want to support additional keybinding magic
#
"base_html_template": "templates/base_template.html",
"header_file": "templates/header.html",

# This is where the actual HTML files will be generated. You should not need to change␣
↪this normally.
"site_dir": "potato/templates/",
```

1.6 User and collaboration

1.6.1 Sign up and log in

Potato supports self-registration by default. A new user can register for the annotation task by navigating to the annotation server, then selecting "Create your account".



After creating an account, you can log in with the email and password used in the account creation step.

1.6.2 Direct login with URL arguement

Potato also supports direct login through URL arguements, for example: http://localhost:8000/?PROLIFIC_PID=user

You could setup direct login in the YAML configuration file (example):

```
#defining the ways annotators entering the annotation system
"login": {
   "type": 'url_direct',    #can be 'password' or 'url_direct'
   "url_argument": 'PROLIFIC_PID' # when the login type is set to 'url_direct', 'url_
↪argument' must be setup for a direct url argument login
},
```

1.6.3 Collaboration under local network

If you do not want to expose the annotation app globally, Potato serves to the local area network by default. You can access the Potato instance through the local IP address of the server.

On Linux machines, you can determine the local IP by running

```
hostname -I
```

1.7 Automatic task assignment

Potato can automatically assign instances to annotators based on figurations.

```
  "automatic_assignment": {
  "on": True, #whether do automatic task assignment for annotators, default False.
  "output_filename": 'task_assignment.json',
  "sampling_strategy": 'random', #currently we support random assignment or ordered␣
↪assignment. Use 'random' for random assignment and 'ordered' for ordered assignment
  "labels_per_instance": 3,  #the number of labels for each instance
  "instance_per_annotator": 5, #the total amount of instances to be assigned to each␣
↪annotator
  "test_question_per_annotator": 0, # the number of attention test question to be␣
↪inserted into the annotation queue. you must set up the test question in surveyflow to␣
↪use this function

  "users": [  ],
},
```

1.8 Productivity features

1.8.1 Keyboard shortcuts

Sequential keybindings: Some annotation schemes provide keybindings for selecting options. For tasks where there are at most 10 options, keybindings can be assigned sequentially by default. When defining your annotation scheme, set the `sequential_key_binding` field to `True`.

The first option will correspond to the "1" key, the second to the "2" key, . . . , the tenth to the "0" key.

Custom keybindings: For greater control, custom keybindings can also be configured. In this case, pass in objects into the `labels` field of the annotation scheme. Each label object can take a `key_value` field specifying the key that corresponds to it.

For example,

```
 "annotation_schemes": [
     {
         "annotation_type": "multiselect",
         "labels": [
             {
               "name": "Option 1",
               "key_value": '1'
             },
             {
               "name": "Option 2",
               "key_value": '2'
              }
          ]
      }
]
```

1.8.2 Dynamic highlighting

Potato also includes randomized keyword highlights to aid in the annotation process. To enable dynamic highlighting, just provide a path to a tab-separated values file of keywords. The keywords file should have this format:

```
Word Label   Schema
good*        Negative        Sentiment
bad* Positive        Sentiment
terrible     Negative        Sentiment
```

Where the values in the Word column can be any valid regex, the value in the i p Label column corresponds to the selection label and the value in the Schema column corresponds to the annotation schema the label is listed under. A single keywords file can support multiple schemas.

Provide the path to the keywords file as the value to the `keyword_highlights_file` key in the configuration file.

There is currently no way to specify the colors used through the configuration file.

1.8.3 Tooltips

For radio and multiselect question types, you have the option to add tooltips with more details about each response option. You can do this in two ways.

Option 1: you can enter plaintext in the `tooltip` field and the unformatted text will display when you hover your mouse over the response option.

```
 "annotation_schemes": [
 {
      "annotation_type": "multiselect",
      "name": "Question",
      "labels": [
          {
            "name": "Label 1",
            "tooltip": "lorem ipsum dolor",
          },
      ]
 },
]
```

Option 2: you can create an HTML file with formatted text (e.g., bold, unordered list), and pass the path to the html file to the `tooltip_file` field. The formatted text will display when you hover your mouse over the response option.

```
 "annotation_schemes": [
 {
      "annotation_type": "multiselect",
      "name": "Question",
      "labels": [
          {
            "name": "Label 1",
            "tooltip_file": "config/tooltips/label1_tooltip.html"
          },
      ]
 },
]
```

1.8.4 Active learning

Active learning can be enabled and configured by providing the `active_learning_config` key to the configuration file. See below for a basic example of the active learning configuration.

```
# This controls whether Potato will actively re-order unlabeled instances
# during annotation after a certain number of items are annotated to
# prioritize those that a basic classifier model is most uncertain about. If
# you have lots of unlabeled data, active learning can potentially help
# maximize the data you get for your "annotation budget", though if you plan
# on annotative *all* the data, active learning will have no effect.
"active_learning_config": {

  "enable_active_learning": True,

  # The fully specified name of an sklearn classifier object with packages,
```

(continues on next page)

(continued from previous page)

```
  # e.g., "sklearn.linear_model.LogisticRegression". This classifier will be
  # trained on the annotated data and used to re-order the remaining
  # instances.
  "classifier_name": "sklearn.linear_model.LogisticRegression",

  # Any kwargs that you want to pass to the classifier during instantiation
  "classifier_kwargs": { },

  # The fully specified name of an sklearn tokenizer object with packages,
  # e.g., "sklearn.feature_extraction.text.CountVectorizer". This tokenizer
  # will be used to tranform the text instances into features.
  "vectorizer_name": "sklearn.feature_extraction.text.CountVectorizer",

  # Any kwargs that you want to pass to the tokenizer during instantiation.
  #
  # NOTE: it's generally a good idea to keep the active learning classifier
  # "fast" so that annotators aren't waiting long when classifying. This
  # often meanings capping the number of features
  "vectorizer_kwargs": { },

  # When multiple annotators have labeled the same data, this option decides
  # how to resolve the mulitple annotations to a single label for the
  # purpose of training the active learning classifier.
  "resolution_strategy": "random",

  # Some part of the data should still be randomly selected (i.e., not based
  # on active learning). This ensure the annotation process can still see a
  # variety of unbiased samples and that the test data can be drawn from an
  # empirical distribution of the data.
  "random_sample_percent": 50,

  # The names of all annotation schema that active learning should be run
  # for. If multiple schema are provided, an instance will be prioritized
  # based on its lowest certainty across all schema (i.e., the
  # least-confident items).
  #
  # NOTE: if this field is left unset, active learning will use all schema.
  "active_learning_schema": [ "favorite_food" ],

  "update_rate": 5,

  "max_inferred_predictions": 20,
},
```

1.8.5 Automatic task assignent

Potato allows you to easily assign annotation tasks to different annotators, this is especially userful for crowdsourcing setting where you only need one annotator to work on a fixed amount of instances.

You can edit the automatic_assignmetn section in the configureation file for this function

```
 "automatic_assignment": {
    "on": true, # set false to turn off automatic assignment
    "output_filename": "task_assignment.json", # saving path of the task assignment␣
↪status
    "sampling_strategy:": "random", # currently we only support random assignment
    "labels_per_instance": 10, # number of labels for each instance
    "instance_per_annotator": 50, # number of instances assigned for each annotator
    "test_question_per_annotator": 2, # number of attention test questions for each␣
↪annotator
    "users": []
},
```

1.9 Surveyflow

Potato allows you to easily set up a series of modules traditionally used in social science surveys.

1.9.1 Pre-screening questions

You could easily insert any survey questions before the annotation instances using our built-in schemas: likert, radio, checkbox, textbox, drop-down list. Potato also provide templates for setting up task instructions and user consents.

Step 1, prepare a .jsonl file for the survey questions you want to insert. For example, if you want to insert a page of censent questions, you can add the following likes to a jsonl file named consent.jsonl

```
{"id":"1","text":"I certify that I am at least 18 years of age.","schema": "radio",
↪"choices": ["I agree", "I disagree"], "label_requirement": {"right_label":["I agree"]}}
{"id":"2","text":"I have read and understood the information above.","schema": "radio",
↪"choices": ["Yes", "No"], "label_requirement": {"right_label":["Yes"]}}
{"id":"3","text":"I understand I might see potentially offensive or sexual content.",
↪"schema": "radio", "choices": ["Yes", "No"], "label_requirement": {"right_label":["Yes
↪"]}}
{"id":"4","text":"I want to participate in this research and continue with the study.",
↪"schema": "radio", "choices": ["Yes", "No"], "label_requirement": {"right_label":["Yes
↪"]}}
```

Step 2, insert the file path into the configuration file:

```
"surveyflow": {
        "on": true,
        "order": [
            "pre_annotation",
            "post_annotation"
        ],
```

(continues on next page)

(continued from previous page)

```
        "pre_annotation": [
            "projects/your-project-name/surveyflow/English/consent.jsonl",
        ],
        "post_annotation": [

        ],
        "testing": [

        ]
},
```

Potato will automatically create a consent page for all the annotators when you launch it.

1.9.2 Pre-study qualification test

Pre study test allow you to filter annotators based on their ability to answer your questions. You can set up a list of questions with groundtruth and potato will automatically generate the qualification test for you. You may check the example prestudy project to get a rough idea of how it works.

1.9.3 Attention test

Potato also allows you to easily assign attention test questions into the annotation instances, just create another jsonl file, for example:

```
{"id":"test_question","text":"This is a test question, please select [test_question_
↪choice].", "choices": ["1", "2", "3", "4", "5"]}
```

and edit the surveyflow section in the configuration file:

```
"surveyflow": {
        "on": true,
        "order": [
            "pre_annotation",
            "post_annotation"
        ],
        "pre_annotation": [
            "projects/your-project-name/surveyflow/consent.jsonl",
        ],
        "post_annotation": [

        ],
        "testing": [
             "projects/your-project-name/surveyflow/English/testing.jsonl",
        ]
},
```

1.9.4 Post-screening questions

You can also insert post study surveys just like the prestudy survey:

```
{"id":"1","text":"What gender do you most closely identify with?","schema": "radio",
↪"choices": ["Male", "Female", "Non-binary"], "label_requirement": {"required":true}}
```

and add the filename into the surveyflow section of your configuration file:

```
"surveyflow": {
        "on": true,
        "order": [
            "pre_annotation",
            "post_annotation"
        ],
        "pre_annotation": [
            "projects/your-project-name/surveyflow/consent.jsonl",
        ],
        "post_annotation": [
            "projects/your-project-name/surveyflow/demographics.jsonl",
        ],
        "testing": [
             "projects/your-project-name/surveyflow/testing.jsonl",
        ]
},
```

1.9.5 Built-in demographic questions

Potato provides a list of basic demographic questions covering common needs:

```
{"id":"1","text":"What gender do you most closely identify with?","schema": "radio",
↪"choices": ["Male", "Female", "Non-binary"], "label_requirement": {"required":true}}
{"id":"2","text":"What is your current age?","schema": "number", "label_requirement": {
↪"required":true}}
{"id":"3","text":"What is your occupation?","schema": "radio", "choices": ["Employed",
↪"Unemployed", "Student", "Retired", "Homemaker", "Self-employed", "Other"], "label_
↪requirement": {"required":true}}
{"id":"4","text":"What is your education level?","schema": "radio", "choices": ["Less␣
↪than a high school diploma", "High school diploma or equivalent", "College degree",
↪"Graduate degree", "Other"], "label_requirement": {"required":true}}
{"id":"5","text":"What is your country of birth?","schema": "select", "use_predefined_
↪labels": "country", "label_requirement": {"required":true}}
{"id":"6","text":"In which country did you spend most of your time before you turned 18?
↪","schema": "select", "use_predefined_labels": "country", "label_requirement": {
↪"required":true}}
{"id":"7","text":"Which country are you currently living in?", "schema": "select", "use_
↪predefined_labels": "country", "label_requirement": {"required":true}}
{"id":"8","text":"What ethnic group do you belong to?","schema": "select", "use_
↪predefined_labels": "ethnicity", "label_requirement": {"required":true}}
{"id":"9","text":"What is your present religion, if any?","schema": "select", "use_
↪predefined_labels": "religion", "label_requirement": {"required":true}}
```

(continues on next page)

(continued from previous page)

```
{"id":"10","text":"Please feel free to leave any comments about our study (optional)",
↪"schema": "text"}
```

1.9.6 Built-in study experience survey:

Potato also supports you to survey the user annotation experience with the following questions:

```
{"id":"1","text":"How satisfied do you feel about your experience participating our␣
↪study?","schema": "radio", "choices": ["Not satisfied", "Satisfied", "Very satisfied"],
↪ "label_requirement": {"required":true}}
{"id":"2","text":"How do you feel about your experience participating our study compared␣
↪with other studies?","schema": "radio", "choices": ["Much worse than others", "Worse␣
↪than others", "Similar", "Better than others", "Much better than others"], "label_
↪requirement": {"required":true}}
{"id":"3","text":"Please feel free to leave any comments about our study (optional)",
↪"schema": "text"}
```

1.10 Crowdsourcing setting

Potato can be seamlessly deployed online to collect annotations from common crowdsourcing platforms like Prolifc.com

1.10.1 Setup potato on a server with open ports

To run potato in a crowdsourcing setup, you need to setup potato on a server with open ports (ports that can be accessed via open internet). When you start the potato server, simply change to default port to the openly accessible ports and you should be able to access the annotation page via you_ip_address:the_port

1.10.2 Prolific

Prolific is a platform where you can easily recruit task participants and Potato can be used seamlessly with prolific.co. To used potato with prolific.co, you need to define the login type as *url_direct* and set up the "url_argument" as 'PROLIFIC_PID'.

```
#defining the ways annotators entering the annotation system
"login": {
   "type": 'url_direct',    #can be 'password' or 'url_direct'
   "url_argument": 'PROLIFIC_PID' # when the login type is set to 'url_direct', 'url_
↪argument' must be setup for a direct url argument login
},
```

It is also recommended to set the "jumping_to_id_disabled" and "hide_navbar" as True

```
#the jumping-to-id function will be disabled if "jumping_to_id_disabled" is True
 "jumping_to_id_disabled": False,

#the navigation bar will be hidden to the annotators if "hide_navbar" is True
 "hide_navbar": True,
```

1.11 Annotator statistics

1.11.1 Inter-annotator-agreement

In development

1.11.2 Annotation time

Potato automatically tracks the time each annotator spent on each instance, which is available in the annotation output file.

```
{"id": "item_11", "annotation": {"awesomeness": {"scale_3": "true"}}, "behavioral_data":
↪{"time_string": "Time spent: 0d 0h 0m 5s "}}
{"id": "item_12", "annotation": {"awesomeness": {"scale_4": "true"}}, "behavioral_data":
↪{"time_string": "Time spent: 0d 0h 0m 1s "}}
{"id": "item_13", "annotation": {"awesomeness": {"scale_4": "true"}}, "behavioral_data":
↪{"time_string": "Time spent: 0d 0h 0m 1s "}}
```

1.12 Open-sourcing

Potato is fully open-sourced and welcome contributions to our project.

1.12.1 Our current development plan

1. better UI setup

1.12.2 Submit feature request

You may submit a feature request in our github repo

1.12.3 Submit pull request

Please contact pedropei@umich.edu to contribute to potato