**DSBA 6188 - Homework 2**
**Manjinder Sandhu**

**Task:** The task for this assignment is to develop a text classifier that can filter through posts and comments on Reddit Cooking Threads and categorize content relevant to recipes. It will not include unrelated discussions and noise. By implementing this classifier, stakeholders will be able to specifically focus on recipe topics and gather insight into cooking trends on Reddit.

## Approach and Reasoning

When we started this assignment we had multiple people label a dataset called "cooking_eval_reviewed2.jsonl" to create a golden evaluation dataset. By having multiple people and myself label this dataset it ensures consistency and prevents any discrepancies that can affect the dataset. This approach reduced errors in the dataset and created a more accurate and trustworthy evaluation. We are going to use this "golden dataset" to evaluate the training model. We used "ngrok" to accomplish this.

In the appendix, there will be the inter-annotator agreement that my teammates came up with. The inter-annotator agreement shows the level of agreement between the annotators by seeing how often the same label was annotated on the same data. This is very important because it allows us to understand the quality of the annotations and ensure the labeled dataset is reliable. ChatGPT was also used to help with coding, understanding concepts, and fixing errors.

The training dataset that I used to label my data was called "homework2_training". This dataset had 5,000 unlabeled posts and comments from Reddit. I labeled 500 posts/comments then I did 2 experiments. The first experiment was with Prodigy Train where on average my evaluation score was 0.74. The second experiment I did was with spacy train and used a based model called "en_core_web_lg", my evaluation score was on average around 0.925. This was done only on 10 percent of the data. I wanted a way where I could label all 5,000 posts/comments without spending too much time. I used a concept called Weakly labels.

Weakly labels is a technique in spaCy where instead of manually labeling your annotations you can label them using heuristic rules, distance supervision, and bootstrapping. These methods allow you to assign labels to large datasets with minimal human effort. Weakly labeling may not be as precise or reliable as manually labeling a dataset. There are several techniques to implement a Weakly label:

1. **Bootstrapping**:
    a. This is where you manually label a small amount of the data and then use a model train on the data to label the rest. I used this method.
2. **Heuristic Rules**:

    a. This is where you establish patterns and rules to automatically label your data. For example, in this dataset, any keywords related to cooking might be weakly labeled as a recipe.
3. **Distance Supervision**:
    a. This is where you use metadata or other information from the database to assign labels. You are using external sources to label the data.
4. **Crowdsourcing**:
    a. This is where you get a team and everyone labels a dataset.

When I applied Weakly label I broke down each step instead of writing one code that does everything. I did this to better apply the concept and fully understand it.

**Step 1:**
I first had to get my 500 labeled data and export it out of the database. I did this by

```
prodigy db-out homework2_manual > exported_data.jsonl
```

After getting those 500 labels, I had to format them in a way that spaCy could read them. I want spaCy to read my annotations so I can train my model. I did this by using the "readSp.py" code and the output file was called "processed_data.jsonl". The processed_data.jsonl files had the labels "RELEVANT" or "NOT_RELEVANT" for each jsonl line.

**Step 2:**
After formatting the jsonl file for spaCy. I used a Python script called "score_records.py" that trained a text classification model to tell the difference between RELEVANT" or "NOT_RELEVANT". It initialized a blank English spaCy model and added a text classification pipeline to it. Then it used stochastic gradient descent (SGD) to train it. Once it is done training it is saved to the local disk for future text classification. You can also find this in the file called "my_trained_model".

**Step 3:**
In "codethatLabel.py", we load the pre-trained classification model using spaCy and we iterate through each JSONL line applying the model to predict the labels. We add the original unlabeled files as the input file. It will give a "RELEVANT" and "NOT_RELEVANT" score. The output for this is "classified_data.jsonl".

**Step 4:**
The last step is to transform the classified data to another format based on the threshold. We can do this by running "finalstep.py" The threshold is 0.5. Based on the threshold it will assign "accept" or "reject" to the record. The code generates an output record for each record. The output file for this is called "homework2_trainComplete.jsonl" with 5000 records annotated.

I will assess my new training data again against my golden dataset. The third experiment was with Prodigy Train where on average my evaluation score was 0.75. The fourth experiment I did was with the spaCy train and used a based model called “en_core_web_lg”, my evaluation score was on average around 0.81. There was a slight difference between training 500 vs 5,000 on the evaluation. I believe that the 500 labeled data were the most straightforward to distinguish. When I used 100 percent of the data, I received a more diverse range of examples that had more challenging data. To fix this I can perform data augmentation and regularization techniques.

There are several files that I did not include in the final GitHub Repositories like the venv, output/experiment-2/model-best/vocab/vectors because those files were over 500 MB. They are in the git ignore.

## Annotated Guidelines for Classification on Recipes

**Task:** The goal of these guidelines is to provide instructions for labeling “RELEVANT” or “NOT_RELEVANT” for specific comments/posts on Reddit Cooking Threads.

Definition:

**RELEVANT:** This is a post/comment on Reddit Cooking Threads that is related to recipes. It can include instructions for preparing a dish or discussing ingredients for a recipe.

**NOT_RELEVANT:** This is a post/comment on Reddit Cooking Threads that isn’t related to recipes. It doesn’t contain any details on how to prepare a dish or doesn’t have any food-related content.

Examples:

✅ It includes 5 pounds of flour and 6 tablespoons of olive oil.
❌It has 4 knives and 2 blenders

✅I like to use flatbread. Naan or Pati. Ensure it is whole grain. For vegetables use tomato, asparagus, artichokes, and peas.
❌I like to cook my flatbread on a stove because it says it in a recipe book.

There are no recipes in these comments, but they are related to discussing ingredients in a dish so the green is RELEVANT while the red is NOT_RELEVANT. The red doesn’t help add anything meaningful to the dish discussion or come up with new recipes.

**Proposal For Recipe Classifier For Different Cuisine Types**

**Objective:** The goal for this is to create a machine-learning model that can classify recipes based on cuisine type. Some of the cruises that we may include are Mexican, Italian, Indian, Mediterranean, and many more.

**Approach:**

Data Collection: We want to gather different datasets that have different recipes from various cuisine types. We are going to look for publicly available datasets or APIs to get this information.

Data Preprocessing: In this step, we will tokenize the recipe into words for further processing. It is important to conduct data augmentation techniques to diversify the dataset. We can perform standardization to remove irrelevant information.

Model Training: In this step, we will use spaCy to implement a multi-class text classification model. It is also good to fine-tune the pre-trained model and I recommend that we use "en_core_web_lg".

Annotations: In this stage, we want to annotate the date efficiently so we will use Prodigy. We can get a group of people to annotate different cuisine types they see in the dataset.

Evaluation: Then we have to assess the model performance by looking at the F1-score. We can also use cross-validation to ensure that the model can perform well on unseen data. If the model performs low then we can go back and review the annotations to further improve the model.

Timeline: It will take our company around 8 months to complete this project. The 8 months will factor in delays and other time-consuming factors that may affect the project.

Teams:

- NLP Engineers
- Data Scientist
- Project Manager

Conclusion:
We are going to provide a high-quality machine-learning model that classifies recipes based on cruising type. We will use tools such as Prodigy and spaCy. This project will take around 8 months to complete.

## Appendix:

Inter-Annotator Agreement:

```
ℹ Using 4 annotator IDs: cooking_eval-Tyler, cooking_eval-alex,
cooking_eval-Manjinder, cooking_eval-"dagim"
ℹ Annotation Statistics

Attribute                      Value
----------------------------   -----
Examples                         800
Categories                         2
Co-Incident Examples*            200
Single Annotation Examples         0
Annotators                         4
Avg. Annotations per Example     4.0

* (>1 annotation)

ℹ Agreement Statistics

Statistic                     Value
--------------------------   ------
Percent (Simple) Agreement   0.7067
Krippendorff's Alpha         0.4075
Gwet's AC2                   0.4197
```

Experiment 1: 500 Prodigy Train

```
=============================== Training pipeline ===============================
Components: textcat_multilabel
Merging training and evaluation data for 1 components
  - [textcat_multilabel] Training: 500 | Evaluation: 200 (from datasets)
Training: 500 | Evaluation: 200
Labels: textcat_multilabel (2)
ℹ Pipeline: ['textcat_multilabel']
ℹ Initial learn rate: 0.001
E    #       LOSS TEXTC...  CATS_SCORE  SCORE 
---  ------  -------------  ----------  ------
  0       0           0.25       34.88    0.35
  0     200          46.12       69.38    0.69
  2     400          27.12       75.93    0.76
  3     600          18.18       73.69    0.74
  4     800          13.67       74.39    0.74
  6    1000           9.80       73.92    0.74
  8    1200           7.61       74.09    0.74
 11    1400           5.59       73.72    0.74
 14    1600           4.41       73.89    0.74
 18    1800           3.39       73.87    0.74
 23    2000           2.57       73.94    0.74
```

Experiment 2: 500 spaCy

```
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
(venv) (venv) Manjinders-MacBook-Pro:homework2 manjinder$ python -m spacy train corpus/config.cfg --paths.train corpus/train.spacy --paths.dev corpus/dev.spacy
ℹ No output directory provided
ℹ Using CPU

=========================== Initializing pipeline ===========================
✔ Initialized pipeline

============================= Training pipeline =============================
ℹ Pipeline: ['tok2vec', 'tagger', 'parser', 'attribute_ruler',
'lemmatizer', 'ner', 'textcat_multilabel']
ℹ Frozen components: ['tagger', 'parser', 'attribute_ruler',
'lemmatizer', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS TEXTC...  CATS_SCORE  SPEED   SCORE 
---  ------  ------------  -------------  ----------  ------  ------
  0       0          0.02           0.22       43.41  6023.70    0.43
  6    1000         38.20          86.59       91.11  7377.56    0.91
 23    2000         21.64           4.06       88.72  7038.52    0.89
 60    3000         28.00           1.04       88.01  7410.59    0.88
100    4000         47.29           1.20       89.06  7822.06    0.89
140    5000          0.02           0.00       91.66  7321.56    0.92
180    6000          0.00           0.00       91.25  6758.01    0.91
221    7000          0.00           0.00       91.18  7797.50    0.91
261    8000          0.00           0.00       91.32  7691.92    0.91
301    9000          0.00           0.00       91.78  7751.39    0.92
341   10000          0.00           0.00       91.60  7629.94    0.92
381   11000          0.00           0.00       91.69  7598.87    0.92
421   12000          0.00           0.00       91.91  7440.61    0.92
462   13000          0.00           0.00       91.82  7333.26    0.92
502   14000          0.00           0.00       91.86  6927.00    0.92
542   15000          0.00           0.00       92.05  7701.79    0.92
582   16000          0.00           0.00       92.19  7587.03    0.92
622   17000          0.00           0.00       92.37  7479.82    0.92
662   18000          0.00           0.00       92.52  7427.78    0.93
703   19000          0.00           0.00       92.64  7509.34    0.93
743   20000          0.00           0.00       92.75  7500.18    0.93
783   21000          0.00           0.00       92.88  7487.14    0.93
824   22000          0.00           0.00       93.02  7453.42    0.93
864   23000          0.00           0.00       93.13  7510.56    0.93
904   24000          0.00           0.00       93.18  7362.93    0.93
944   25000          0.00           0.00       93.15  7459.23    0.93
985   26000          0.00           0.00       93.16  7430.85    0.93
1025   27000          0.00           0.00       93.13  7595.56    0.93
1065   28000          0.00           0.00       93.08  7432.19    0.93
1105   29000          0.00           0.00       93.11  7498.57    0.93
```

Experiment 3: 5000 Prodigy Train

```
Training: 5000 | Evaluation: 200
Labels: textcat_multilabel (2)
ℹ Pipeline: ['textcat_multilabel']
ℹ Initial learn rate: 0.001
E    #       LOSS TEXTC...  CATS_SCORE  SCORE 
---  ------  -------------  ----------  ------
  0       0           0.25       32.27    0.32
  0     200          48.93       54.73    0.55
  0     400          34.34       65.12    0.65
  0     600          35.86       70.19    0.70
  0     800          33.11       72.95    0.73
  0    1000          29.71       73.90    0.74
  0    1200          27.21       73.30    0.73
  1    1400          23.50       72.37    0.72
  1    1600          21.05       72.91    0.73
  1    1800          20.14       73.42    0.73
  2    2000          18.57       73.68    0.74
  3    2200          16.78       74.70    0.75
  3    2400          14.63       74.31    0.74
  4    2600          13.19       74.85    0.75
  5    2800          12.20       75.02    0.75
  6    3000          11.32       75.53    0.76
  7    3200          10.05       74.95    0.75
  7    3400           8.89       75.09    0.75
  8    3600           8.72       75.09    0.75
  9    3800           7.60       74.91    0.75
 10    4000           7.26       75.41    0.75
 11    4200           6.62       75.38    0.75
 12    4400           6.12       75.04    0.75
 12    4600           6.05       75.31    0.75
```

Experiment 4: 5000 spaCy

```
============================= Generating config =============================
ℹ Auto-generating config with spaCy
ℹ Using config from base model
✔ Generated training config

========================= Generating cached label data =========================
✔ Saving label data for component 'tagger'
corpus/labels/tagger.json
✔ Saving label data for component 'parser'
corpus/labels/parser.json
✔ Saving label data for component 'ner'
corpus/labels/ner.json
✔ Saving label data for component 'textcat_multilabel'
corpus/labels/textcat_multilabel.json

============================= 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
(venv) (venv) Manjinders-MacBook-Pro:homework2 manjinder$ python -m spacy train corpus/config.cfg --paths.train corpus/train.spacy --paths.dev corpus/dev.spacy
ℹ No output directory provided
ℹ Using CPU

=========================== Initializing pipeline ===========================
✔ Initialized pipeline

============================= Training pipeline =============================
ℹ Pipeline: ['tok2vec', 'tagger', 'parser', 'attribute_ruler',
'lemmatizer', 'ner', 'textcat_multilabel']
ℹ Frozen components: ['tagger', 'parser', 'attribute_ruler',
'lemmatizer', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS TEXTC...  CATS_SCORE  SPEED   SCORE 
---  ------  ------------  -------------  ----------  ------  ------
  0       0          0.09           0.29       40.33 4139.42    0.40
  0    1000         34.09         179.79       87.48 7990.48    0.87
  2    2000         51.11         115.09       84.23 7915.07    0.84
  6    3000         99.28          37.74       81.08 6832.15    0.81
 10    4000         85.82          13.12       80.13 6611.60    0.80
 14    5000         86.32           7.52       80.76 7223.10    0.81
 18    6000         78.08           4.43       80.30 7075.75    0.80
```

**Commands to run this file:**

WARNING: Ensure that you have a Prodigy License. All these Command Lines were done in the terminal.

`source venv/bin/activate`

`python3 -m prodigy textcat.manual homework2_manual data/homework2_train.jsonl --label RELEVANT,NOT_RELEVANT`

```
(venv) Manjinders-MacBook-Pro:homework2 manjinder$ source venv/bin/activate
(venv) (venv) Manjinders-MacBook-Pro:homework2 manjinder$ python3 -m prodigy textcat.manual homework2_manual data/homework2_train.jsonl --label RELEVANT,NOT_RELEVANT
Using 2 label(s): RELEVANT, NOT_RELEVANT

✨  Starting the web server at http://localhost:8080 ...
Open the app in your browser and start annotating!
```

`python3 -m prodigy train --textcat-multilabel homework2_manual,eval:hmwk2-eval ./output/experiment-1`

```
============================= Training pipeline =============================
Components: textcat_multilabel
Merging training and evaluation data for 1 components
  - [textcat_multilabel] Training: 500 | Evaluation: 200 (from datasets)
Training: 500 | Evaluation: 200
Labels: textcat_multilabel (2)
ℹ Pipeline: ['textcat_multilabel']
ℹ Initial learn rate: 0.001
E    #       LOSS TEXTC...  CATS_SCORE  SCORE
---  ------  -------------  ----------  ------
  0       0           0.25       34.88    0.35
  0     200          46.12       69.38    0.69
  2     400          27.12       75.93    0.76
  3     600          18.18       73.69    0.74
  4     800          13.67       74.39    0.74
  6    1000           9.80       73.92    0.74
  8    1200           7.61       74.09    0.74
 11    1400           5.59       73.72    0.74
 14    1600           4.41       73.89    0.74
 18    1800           3.39       73.87    0.74
 23    2000           2.57       73.94    0.74
```

`python3 -m prodigy data-to-spacy --textcat-multilabel homework2_manual,eval:hmwk2-eval ./corpus --base-model en_core_web_lg`

`python -m spacy train corpus/config.cfg --paths.train corpus/train.spacy --paths.dev corpus/dev.spacy`

```
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
(venv) (venv) Manjinders-MacBook-Pro:homework2 manjinder$ python -m spacy train corpus/config.cfg --paths.train corpus/train.spacy --paths.dev corpus/dev.spacy
ℹ No output directory provided
ℹ Using CPU

=========================== Initializing pipeline ===========================
✔ Initialized pipeline

============================= Training pipeline =============================
ℹ Pipeline: ['tok2vec', 'tagger', 'parser', 'attribute_ruler',
'lemmatizer', 'ner', 'textcat_multilabel']
ℹ Frozen components: ['tagger', 'parser', 'attribute_ruler',
'lemmatizer', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS TEXTC...  CATS_SCORE  SPEED   SCORE
---  ------  ------------  -------------  ----------  ------  ------
  0       0          0.02           0.22       43.41  6023.70    0.43
  6    1000         38.20          86.59       91.11  7377.56    0.91
 23    2000         21.64           4.06       88.72  7038.52    0.89
 60    3000         28.00           1.04       88.01  7410.59    0.88
100    4000         47.29           1.20       89.06  7822.06    0.89
140    5000          0.02           0.00       91.66  7321.56    0.92
180    6000          0.00           0.00       91.25  6758.01    0.91
221    7000          0.00           0.00       91.18  7797.50    0.91
261    8000          0.00           0.00       91.32  7691.92    0.91
301    9000          0.00           0.00       91.78  7751.39    0.92
341   10000          0.00           0.00       91.60  7629.94    0.92
381   11000          0.00           0.00       91.69  7598.87    0.92
421   12000          0.00           0.00       91.91  7440.61    0.92
462   13000          0.00           0.00       91.82  7333.26    0.92
502   14000          0.00           0.00       91.86  6927.00    0.92
542   15000          0.00           0.00       92.05  7701.79    0.92
582   16000          0.00           0.00       92.19  7587.03    0.92
622   17000          0.00           0.00       92.37  7479.82    0.92
662   18000          0.00           0.00       92.52  7427.78    0.93
703   19000          0.00           0.00       92.64  7509.34    0.93
743   20000          0.00           0.00       92.75  7500.18    0.93
783   21000          0.00           0.00       92.88  7487.14    0.93
824   22000          0.00           0.00       93.02  7453.42    0.93
864   23000          0.00           0.00       93.13  7510.56    0.93
904   24000          0.00           0.00       93.18  7362.93    0.93
944   25000          0.00           0.00       93.15  7459.23    0.93
985   26000          0.00           0.00       93.16  7430.85    0.93
1025   27000           0.00           0.00       93.13  7595.56    0.93
1065   28000           0.00           0.00       93.08  7432.19    0.93
1105   29000           0.00           0.00       93.11  7498.57    0.93
```

prodigy db-out homework2_manual > exported_data.jsonl


{"text":"A ss pan isn't going to hold it's heat as well as cast iron. If you want the best browning, it's gotta be cast iron. If you'
{"text":"Your sauce recipe has a lot of starch for the amount of liquid. It's a wonder that you got any sauce at all. Also that's a lo
{"text":"I like to dip saltines into Lipton tea that has sugar in it. Weird, I know, but it\u2019s something I did as a kid and still
{"text":"I've said \"behind\" so often I use it In super markets and the likes.","meta":{"source":"homework2-DSBA6188-UNCC"},"_input_h
{"text":"Do they? I know very little about cooking curries, just that they're delicious. Can you recommend good resources for learning
{"text":"> downvoted, at the bottom of the page. I think you were in Australia at the time of typing.","meta":{"source":"homework2-DSE
{"text":"This is really good: Smoky Lentil Chili https://www.lemonsforlulu.com/smoky-lentil-chili/","meta":{"source":"homework2-DSBA61
{"text":"the threads can sometimes indicate that it's very dry. looks to be the case here. It's possible that if this was frozen previ
{"text":"That's basically what I did :) I had been eating my bacon as I was making the pancakes, so just crumbled a piece on top.","me
{"text":"They can be used to make a pretty decent sauce for a pork tenderloin. Or they can be used reduced down and used to make a BBQ
{"text":"That makes a lot of sense compared to other meats like pork and beef. I usually only see leg, chops, and rack of lamb in the
{"text":"Given the state of research on diet, it's hard to say how much you should care about the recipe using butter vs olive oil, bu
{"text":"How the hell did they chip it up like that cutting a pizza?!? In any case, ouch!! I guess its time to send it to Shun for sha
{"text":"In my opinion, unless your Vietnamese Grandma is gonna cook all day, the best way to eat pho is at a restaurant (hint: They a
{"text":"Check out the handles on All-Clad. Some people hate the handles. Other are fine with them. The handles dig into my palm in a
{"text":"Mix equal parts olive oil, balsamic vinegar, and some sort of mustard (I go with Dijon) and dip with bread. 7/7","meta":{"sou
{"text":"Not gonna lie. Shoyu just doesn't do it for me no more.","meta":{"source":"homework2-DSBA6188-UNCC"},"_input_hash":-701354210
{"text":"I tried so many all-butter chocolate chip cookie recipes because I assumed butter was just a superior ingredient to use, beca
{"text":"I am a bit old fashioned. If i find a good recipe that i want to keep, then i write it in my recipe book. Most things you don

python3 readSp.py

```python
import jsonlines

# Path to your dataset file
dataset_file = "exported_data.jsonl"

# Path to the output file
output_file = "processed_data.jsonl"

# Open the JSONL file and extract text and labels
try:
    with jsonlines.open(dataset_file) as reader, jsonlines.open(output_file, mode='w') as writer:
        for obj in reader:
            text = obj.get("text")
            label = obj.get("accept", [])[0]  # Get the first accepted label if available
            if text and label:
                writer.write({"text": text, "label": label})
            else:
                print("Warning: Text or label missing in the JSON object.")
        print("Processing completed. Output written to:", output_file)
except Exception as e:
    print("Error:", e)

```

python3 score_records.py

score_records.py > ...

```python
import spacy
from spacy.training import Example
import jsonlines
import random
# Load a blank English model
nlp = spacy.blank("en")
# Add text classification pipeline to the model
textcat = nlp.add_pipe('textcat_multilabel', last=True)
textcat.add_label("RELEVANT")
textcat.add_label("NOT_RELEVANT")

# Path to the processed data file
processed_data_file = "processed_data.jsonl"

# Open the JSONL file and extract text and labels
with jsonlines.open(processed_data_file) as reader:
    processed_data = list(reader)

# Convert processed data to spaCy format
spacy_train_data = []
for obj in processed_data:
    text = obj["text"]
    label = {"RELEVANT": obj["label"] == "RELEVANT", "NOT_RELEVANT": obj["label"] == "NOT_RELEVANT"}
    spacy_train_data.append(Example.from_dict(nlp.make_doc(text), {"cats": label}))

# Initialize the model and get the optimizer
optimizer = nlp.initialize()

# Train the text classification model
n_iter = 10
for i in range(n_iter):
    spacy.util.fix_random_seed(1)
    random.shuffle(spacy_train_data)
    losses = {}
    for batch in spacy.util.minibatch(spacy_train_data, size=8):
        nlp.update(batch, losses=losses, sgd=optimizer)
    print("Iteration:", i, "Losses:", losses)

# Save the trained model
output_dir = "./my_trained_model"
nlp.to_disk(output_dir)
```

python3 codethatLabel.py

```python
import spacy
import jsonlines

# Load the trained model
model_path = "./my_trained_model"
nlp = spacy.load(model_path)

# Load the unlabeled data
unlabeled_data_file = "data/homework2_train.jsonl"

# Open the JSONL file and classify each record
classified_data = []
with jsonlines.open(unlabeled_data_file) as reader:
    for record in reader:
        text = record["text"]
        doc = nlp(text)
        predicted_labels = doc.cats
        classified_data.append({"text": text, "predicted_labels": predicted_labels})

# Optionally, you can save the classified data to a file or process it further
output_file = "classified_data.jsonl"
with jsonlines.open(output_file, mode="w") as writer:
    writer.write_all(classified_data)
```

prodigy db-in homework2_manual2 homework_trainComplete.jsonl

python3 -m prodigy textcat.manual homework2_manual2 homework2_trainComplete.jsonl --label RELEVANT,NOT_RELEVANT --exclusive

```
  [textcat_multilabel] Training: 5000 | Evaluation: 200 (from datasets)
Training: 5000 | Evaluation: 200
Labels: textcat_multilabel (2)
ℹ Pipeline: ['textcat_multilabel']
ℹ Initial learn rate: 0.001
E    #       LOSS TEXTC...  CATS_SCORE  SCORE
---  ------  -------------  ----------  ------
  0       0           0.25       32.27    0.32
  0     200          48.93       54.73    0.55
  0     400          34.34       65.12    0.65
  0     600          35.86       70.19    0.70
  0     800          33.11       72.95    0.73
  0    1000          29.71       73.90    0.74
  0    1200          27.21       73.30    0.73
  1    1400          23.50       72.37    0.72
  1    1600          21.05       72.91    0.73
  1    1800          20.14       73.42    0.73
  2    2000          18.57       73.68    0.74
  3    2200          16.78       74.70    0.75
  3    2400          14.63       74.31    0.74
  4    2600          13.19       74.85    0.75
  5    2800          12.20       75.02    0.75
  6    3000          11.32       75.53    0.76
  7    3200          10.05       74.95    0.75
  7    3400           8.89       75.09    0.75
  8    3600           8.72       75.09    0.75
  9    3800           7.60       74.91    0.75
 10    4000           7.26       75.41    0.75
 11    4200           6.62       75.38    0.75
 12    4400           6.12       75.04    0.75
 12    4600           6.05       75.31    0.75
✔ Saved pipeline to output directory
```

python3 -m prodigy data-to-spacy --textcat-multilabel homework2_manual2,eval:hmwk2-eval ./corpus --base-model en_core_web_lg

python -m spacy train corpus/config.cfg --paths.train corpus/train.spacy --paths.dev corpus/dev.spacy

```
============================ Generating config ============================
ℹ Auto-generating config with spaCy
ℹ Using config from base model
✔ Generated training config

======================= Generating cached label data =======================
✔ Saving label data for component 'tagger'
corpus/labels/tagger.json
✔ Saving label data for component 'parser'
corpus/labels/parser.json
✔ Saving label data for component 'ner'
corpus/labels/ner.json
✔ Saving label data for component 'textcat_multilabel'
corpus/labels/textcat_multilabel.json

============================ 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
(venv) (venv) Manjinders-MacBook-Pro:homework2 manjinder$ python -m spacy train corpus/config.cfg --paths.train corpus/train.spacy --paths.dev corpus/dev.spacy
ℹ No output directory provided
ℹ Using CPU

========================== Initializing pipeline ==========================
✔ Initialized pipeline

============================ Training pipeline ============================
ℹ Pipeline: ['tok2vec', 'tagger', 'parser', 'attribute_ruler',
'lemmatizer', 'ner', 'textcat_multilabel']
ℹ Frozen components: ['tagger', 'parser', 'attribute_ruler',
'lemmatizer', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS TEXTC...  CATS_SCORE  SPEED   SCORE
---  ------  ------------  -------------  ----------  ------  ------
  0       0          0.09           0.29       40.33 4139.42    0.40
  0    1000         34.09         179.79       87.48 7990.48    0.87
  2    2000         51.11         115.09       84.23 7915.07    0.84
  6    3000         99.28          37.74       81.08 6832.15    0.81
 10    4000         85.82          13.12       80.13 6611.60    0.80
 14    5000         86.32           7.52       80.76 7223.10    0.81
 18    6000         78.08           4.43       80.30 7075.75    0.80
```