

## Overview

We ran five experiments, explained below. See the appendix for the results and the accompanying Jupyter notebook for reproducibility.

### 1. Develop using GPT3.5 zeroshot

We developed our first model using `gpt3-5-zeroshot.jsonl`, loading it with `db-in` and named it `hmkw_1_zeroshot`. We opted to use `--training patience=800` to reduce the early stopping criteria to improve the time to train.

For this initial model, We did not have a dedicated hold out dataset, and used Prodigy's automatic partitioning of 20% the `hmkw_1_zeroshot` as the evaluation dataset. The model attained **32%** F1 score.

### 2. Create 200 unlabeled for evaluation dataset

For the 2nd experiment, We wanted to provide a dedicated hold-out evaluation dataset. We used a previously annotated dataset that was a `.jsonl` file and loaded it with `db-in` and naming it `hmkw_1_eval`. We then retrain on `hmkw-1-zeroshot` and eval on `hmkw-1-eval`. The model attained **36%** F1 score. We also ran a train-curve, which provided evidence that more annotated data would likely improve the model's performance.

### 3. Train with Workshop Data with model in loop

For the next two experiments, we'll explore: quantity vs. quality.

In experiment 3, we'll use all of the workshop data without correcting it (i.e., quantity). We loaded the data with `db-in`, named it `hmkw-1-workshop`, and merged it with the GPT3.5 Zero Shot as `hmkw_1_train_exp3`. We kept `hmkw_1_eval` as the eval dataset. The model attained **52%** F1 score; therefore adding the workshop dataset adds **+16%** F1 score.

### 4. Run model-as-annotator on Workshop data to correct a sample

In experiment 4, we'll focus on quality, correcting ~200 workshop annotations. We'll use `review` so we can view both the model-predicted and workshop annotations. You could use `ner.manual` (original annotations) or `ner.correct` model predicted alternatively. Merging the corrected annotations with the GPT3.5 Zero Shot, yielded the `hmkw_1_train_exp4`. We'll keep using `hmkw_1_eval` as the evaluation dataset.

The model attained **38%** F1 score. This indicates that while the full workshop dataset may have errors, it provides net gain in performance (quantity over quality).

## 5. Use word vectors (transfer learning) to improve model performance

For experiment 5, we'll explore whether adding word embeddings improves model performance (see [Prodigy docs](#)). We used experiment 3 training dataset (best performance) and use `hmk_1_eval` as the eval dataset. The model attained **59%** F1 score, which is the best performing model/experiment.

## Appendix

### Experiment Performances

#	Training Dataset	Training (n)	Evaluation Dataset	Base Model	F1	P	R
1	GPT3.5 ZShot	400	20% of training	None	0.32	0.44	0.24
2	GPT3.5 ZShot	500	Hold out eval	None	0.36	0.41	0.31
3	GPT3.5 ZShot + Workshop (raw)	1577	Hold out eval	None	0.52	0.56	0.48
4	GPT3.5 ZShot + Workshop Corrected	699	Hold out eval	None	0.38	0.44	0.34
5	GPT3.5 ZShot + Workshop (raw)	1577	Hold out eval	en_core_web_1.5	0.59	0.58	0.59

### Saving Models (wheel)

To export each model, we used `spacy package`, saving it as a Python package to a `models` folder. This saves the model as `wheel` which compressed the model.

To install any of the models, you can the use `pip install` like

```
$ python -m pip install models/en_ner_reddit_cooking-5.0.0.tar.gz
Processing ./models/en_ner_reddit_cooking-5.0.0.tar.gz
  Preparing metadata (setup.py) ... done
...
Successfully built en_ner_reddit_cooking
Installing collected packages: en-ner-reddit-cooking
Successfully installed en-ner-reddit-cooking-5.0.0
```

```
import spacy
nlp = spacy.load("en_ner_reddit_cooking")
doc = nlp("Make a dressing with mayo, plain yogurt, curry powder, mustard powder, mustard,
doc.ents
# (mayo, plain yogurt, curry powder, mustard powder, mustard, ground coriander)
```