HOMEWORK-2: TEXT CLASSIFIER MODEL FOR RECIPES
Group 2: EMBEDDING EAGLES

Modeling methodology:

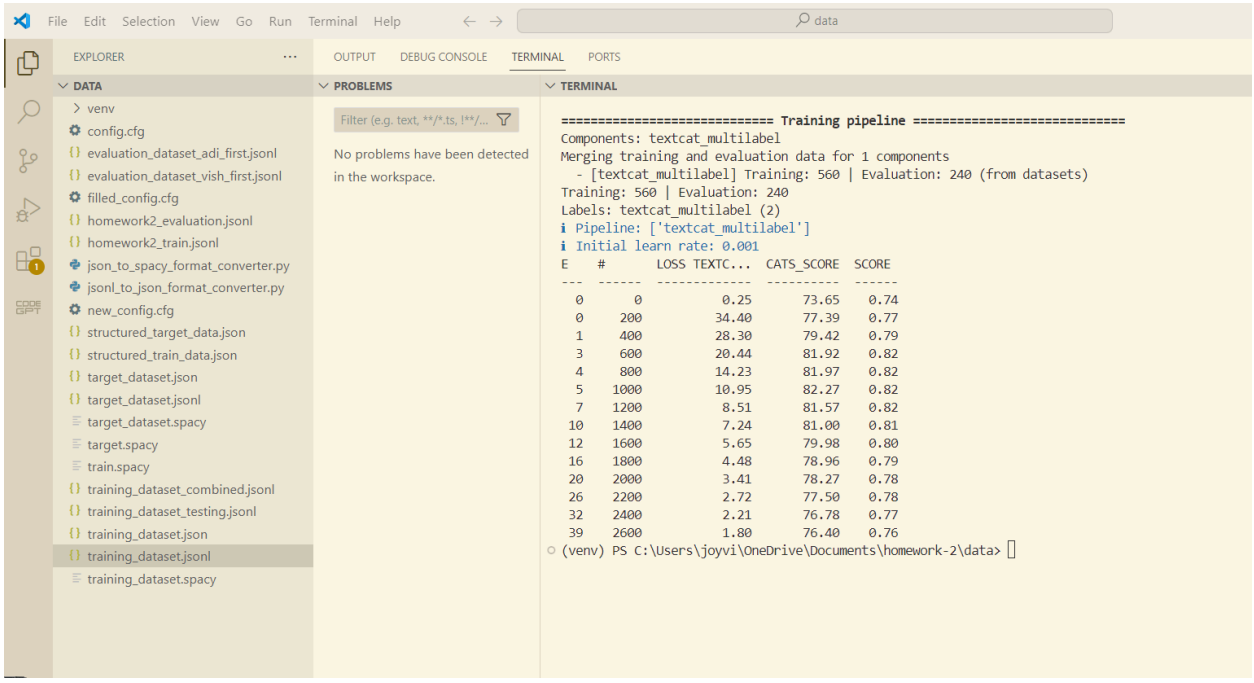
- First, after developing the annotation guidelines, we annotated the evaluation dataset together as a group with each member annotating 200 labels and merging them into a single dataset of 800 entries after exporting the annotations into a JSONL file,
- Then we reviewed each other's annotations and decided on tough labels to compile the knowledge we have got and built a golden evaluation dataset with less noise data and highly accurate labels. After getting the dataset, we used **Cohen's Kappa** module in Python to get metrics for IAA agreement after initial discussion and setup.
- Then, we annotated 800 entries from the training dataset to get the training data and target data ready for the model. We used annotation guidelines to label 800 data entries via group annotation. We repeated the same after obtaining the training dataset to modify and finalize the metrics by considering both dataset data.
- We then split the trained labels dataset of 800 values into training and testing in the ratio of 70% and 30% for constructing the model. We then constructed the model in Prodigy train after trying the model with Spacy approach.
- First, we went with constructing a model using the **Spacy approach** as we are familiar with it. We converted the JSONL datasets (training and target data) into JSON files using Python code and then we converted the JSON files into the spacy model files using the Python code.
- Both the Spacy model files are passed to the" config. cfg" file as training and target models to generate a Text Classifier model using the Spacy approach.
- We built the model and everything went fine but unfortunately we did not get any values in last two columns in the model output. So we switched to **prodigy-train**.
- For the prodigy approach, we used the slides, and video as references to construct the model.
- For prodigy train approach only training data and target data are required. "Config. cfg" file is not required unlike the spacy approach. We used the same training dataset (560 entries) and target dataset (240 entries) from the previous approach.
- First we imported both datasets as train and target datasets to prodigy using "db-in" command.
- After importing datasets to prodigy, we constructed model using "python -m prodigy train --textcat-multilabel training_dataset,eval:target_dataset" command.
- This time we got the model perfectly with around 82% accuracy.
- Then we divided annotated as a group using Prodigy Sessions. Then we saved each other annotations in separate datasets. Then we exported them and tried to review each other annotations for building ultimate golden set but due to "invalid arguments for other three datasets" we were unable to do it inside prodigy. We tried merging all four datasets into one dataset and tried same but we got the same result.
- Instead we met physically discussed annotations and constructed the golden dataset. We provided all the **Inter Annotator Agreement details** in the IAA report. We never used the evaluation dataset in any training to avoid **overfitting mechanism** in the model.

```
PS C:\Users\joyvi\OneDrive\Documents\homework-2\data> .\venv\Scripts\activate
(venv) PS C:\Users\joyvi\OneDrive\Documents\homework-2\data> python -m prodigy db-in train C:\Users\joyvi\OneDrive\Documents\homework
2\data\training_dataset.jsonl
✓ Created unstructured dataset 'train' in database SQLite
✓ Imported 560 annotated examples and saved them to 'train' (session
2024-03-16_15-48-25) in database SQLite
Found and keeping existing "answer" in 560 examples
(venv) PS C:\Users\joyvi\OneDrive\Documents\homework-2\data> python -m prodigy db-in test C:\Users\joyvi\OneDrive\Documents\homework
\data\target_dataset.jsonl
✓ Created unstructured dataset 'test' in database SQLite
✓ Imported 240 annotated examples and saved them to 'test' (session
2024-03-16_15-50-31) in database SQLite
Found and keeping existing "answer" in 240 examples
(venv) PS C:\Users\joyvi\OneDrive\Documents\homework-2\data> python -m prodigy train --textcat-multilabel train,eval:test
ℹ Using CPU

========================= Generating Prodigy config ==========================
ℹ Auto-generating config with spaCy
✓ Generated training config

============================ Initializing pipeline ============================
[2024-03-16 15:52:06,401] [INFO] Set up nlp object from config
Components: textcat_multilabel
Merging training and evaluation data for 1 components
  - [textcat_multilabel] Training: 560 | Evaluation: 240 (from datasets)
Training: 560 | Evaluation: 240
Labels: textcat_multilabel (2)
[2024-03-16 15:52:06,497] [INFO] Pipeline: ['textcat_multilabel']
[2024-03-16 15:52:06,497] [INFO] Created vocabulary
[2024-03-16 15:52:06,497] [INFO] Finished initializing nlp object
[2024-03-16 15:52:07,126] [INFO] Initialized pipeline components: ['textcat_multilabel']
✓ Initialized pipeline

============================== Training pipeline ==============================
Components: textcat_multilabel
Merging training and evaluation data for 1 components
  - [textcat_multilabel] Training: 560 | Evaluation: 240 (from datasets)
Training: 560 | Evaluation: 240
Labels: textcat_multilabel (2)
```



Analysis:

Model Overview

Type: Text Classification (Multi-label)

Components: textcat_multilabel

Training Data: 560 samples (70%)

Evaluation Data: 240 samples (30%)

Labels: 2(RELEVANT, IRRELEVANT)
Learning Rate: Initial 0.001
Training Progress
The training process is described over training data with 560 entries and evaluating with 240 enties, showing a reduction in loss and changes in score metrics. Notably, the model performance improves significantly up to a certain point (1000 iterations) with an accuracy peak around 82.27%, after which the performance starts to decline slightly, indicating potential overfitting or diminishing returns on further training. Overall model scores 82% accuracy and stands as better model without any errors.

Inference (How to Apply)

- Take unlabelled dataset
- Start annotations and export the dataset
- Split datasets into two datasets: training dataset and evaluation dataset
- Import datasets into prodigy
- Train the model using the training dataset
- Pass the evaluation data through the model to get predictions.
- Use the output scores to determine label relevance for each recipe.

Advantages

- Specificity: Tailored for multi-label classification, making it suitable for recipes that can belong to multiple categories.
- Adaptability: Can be fine-tuned or retrained with more data or for similar tasks.
- Efficiency: Relatively quick inference, assuming the model isn't overly complex.
- Overfitting: The performance dip after a peak suggests potential overfitting to the training data. But that is not this case in this scenarios as there is no leakage of data as we never used evaluation dataset at all in training.

Disadvantages

- Data Dependency: The model's effectiveness is heavily reliant on the quality and quantity of the training data.
- Generalization: May not generalize well to drastically different text or recipes outside the training distribution.

Conclusion

The model shows promising results for multi-label text classification with a focused application in recipe relevance. Its performance indicates it has learned to categorize recipes effectively, though care must be taken to manage overfitting but in this scenario, there is no chance of overfitting as evaluation dataset does not involved in training at any chance. Future improvements might include expanding the dataset, incorporating regularization techniques, or experimenting with different architectures to enhance generalization and maintain high performance. Model accuracy is better but that should not be dead end. It can be trained again and again with new examples to get higher accuracy. Every model requires constant training and maintenance to give best results in the long run.