Food Hazard Detection Challenge: SemEval 2025 Task 9

Repository Overview

This repository contains multiple Jupyter notebooks designed to address the **SemEval 2025 Task 9: The Food Hazard Detection Challenge**, hosted on Codalab. The challenge involves predicting hazard and product categories (Task 1) and identifying hazard and product labels (Task 2) using textual input features.

Repository Structure

1. **Notebooks**:
    - The repository includes 8 notebooks, with each one focusing on specific tasks and methodologies.
    - Notebooks prefixed with st1 tackle Task 1 (hazard category and product category label predictions).
    - Notebooks prefixed with st2 address Task 2 (hazard and product label predictions).
    - The second part of the Notebook's name specifies the input type (title or text) which is used for predictions.
    - The third part of the Notebook's name indicates the type of the model used in the Notebook (dense for dense Neural Networks or lstm for LSTM Neural Networks).
2. **CSV files:**
    - Incidents_train.csv : containing the training data (input features and output labels)
    - Incidents_val.csv : containing only the input features of validation data where the outputs should be predicted.
    - Csv files that are named after a Notebook and contain the predictions of the validation data originating from the aforementioned Jupyter file.

Methodology

The notebooks follow a **function-based programming approach**, allowing minimal modifications across different tasks. The overall workflow is consistent across all notebooks, with the following structure:

- An initial cell that imports all the necessary supportive libraries that are used throughout all tasks.
- A function that inputs the column title or the column text from the training data and generates a TFIDF representation of the textual input.
- A function that locates all the different labels in a given label column, e.g. it receives as input the label hazard category and it isolates all the unique hazard category labels.
- A function that receives as input the previously mentioned unique labels and creates their one-hot encoded representation.
- A function that generates the Y part of the dataset transforming the labels into a format that can be used to train machine learning models.

- A function that compiles a dense neural network or an lstm neural network depending on the notebook's name.
- An error report function for a given model.
- And finally a function that converts the model output back to textual format.

How to use this repository

- At first ensure that all the different libraries are installed in your environment. If Jupyter notebooks are used the libraries can be installed using the following command: !pip install pandas numpy scikit-learn tensorflow seaborn matplotlib.
- Following this, download or clone any one of the notebooks (depending on which task, which input and which method you prefer to use) and the csv files incidents_train.csv, incidents_val.csv.
- Ensure that the notebook and the csv files are located in the same folder.
- Then you can run all the cells in the notebook sequentially starting from the first cell.
- In the end a csv file will be produced and it will contain the predictions corresponding to certain labels of the incidents_val file (hazard category and product category labels for st1 notebooks or hazard and product for st2 notebooks).

Benchmark analysis

- All the models where tested separately with the feature title as input and with the feature text as input, hence the difference between their names.
- Moreover, a separate model was trained to predict each label. For instance, in st1 files a model named model_1 was trained to predict hazard-category labels while a separate model named model_2 was trained to predict product-category labels.
- To further elaborate this method was chosen for simplicity. A multi-task learning approach with a neural network that predicts both output labels simultaneously is recommended for further improving performance. This could be achieved if the network was comprised of two separate output layers and a shared hidden layer body that shares the same weights for both tasks.
- In the current state the basic machine learning models used are the dense neural networks, while the more advanced models are the lstm nets.
- The TFIDF vectorizer was used in order to transform the textual inputs into numeric vectors. To be more precise, the TFIDF vectorizer parameters that were chosen were the default ones (unigram, 'word', lowercase). However, a Word2Vec approach was also used but did not eventually remain in the notebooks as for the current setting it hindered the model's performance.
- To evaluate which model and input pair would yield better predictions for the incidents_val data, the incidents_train data were spilt into training and 'test' data. Consequently, the model's performance on predicting the 'test' data along with it's computational complexity

were the main factors considered in order to select the appropriate model and input pair (e.g. dense nn and text or lstm and title).

- To further explain, the metrics that were used for this evaluation were the validation accuracy and the weighted F1 score regarding the 'test' data.
- The weighted F1 score was chosen due to the considerable imbalance of the classes representation in the training set.
- In general, weighted F1 score and validation accuracy followed a similar trend for a given model-input pair.

The performance summary for the different model-input pairs is presented in the table below. In all cases the hyperparameters initial learning rate, optimizer, mini-batches remained equal to 0.001, Adam, 32 respectively. The cost function that was optimized was the Categorical-cross-entropy. Last but not least, the train-test split ratio was 80%-20% and the timesteps of the lstm were 1.

Label	Validation accuracy	F1 score	Input	Model	Architecture	regularization	Epochs
hazard-category	0.83	0.82	title	dense	256(relu),128(relu),10(softmax)	-	7
hazard-category	0.79	0.78	title	lstm	64(Lstm, relu), 32 (Lstm, relu), 64(dense,relu),10(dense,softmax)	0.2 Dropout in Lstm layers	30
hazard-category	0.89	0.88	text	dense	256(relu),120(relu),10(softmax)	-	3
hazard-category	0.86	0.86	text	lstm	64(Lstm, relu), 32 (Lstm, relu), 64(dense,relu),10(dense,softmax)	0.2 Dropout in Lstm layers	15
product-category	0.71	0.71	title	dense	256(relu),128(relu),22(softmax)	-	7
product-category	0.62	0.12	title	lstm	64(Lstm, relu), 32 (Lstm, relu), 64(dense,relu),22(dense,softmax)	0.2 Dropout in Lstm layers	30
product-category	0.72	0.71	text	dense	256(relu),128(relu),22(softmax)	-	4
product-category	0.59	0.6	text	lstm	64(Lstm, relu), 32 (Lstm, relu), 64(dense,relu),22(dense,softmax)	0.2 Dropout in Lstm layers	15
hazard	0.59	0.58	title	dense	256(relu),120(relu),128(softmax)	-	10
hazard	0.47	0.46	title	lstm	256(Lstm, relu), 128 (Lstm, relu), 256(dense,relu),128(dense,softmax)	0.2 Dropout in Lstm layers	10
hazard	0.6	0.6	text	dense	21(relu),10(relu),128(softmax)	-	30
hazard	0.6	0.6	text	lstm	64(Lstm, relu), 32 (Lstm, relu), 64(dense,relu),128(dense,softmax)	0.2 Dropout in Lstm layers	25
product	0.35	0.34	title	dense	256(relu),128(relu),1022(softmax)	-	15
product	0.22	0.0	title	lstm	256(Lstm, relu), 128 (Lstm, relu), 256(dense,relu),1022(dense,softmax)	-	50

product	0.31	0.30	text	dense	256(relu),128(relu),1022(softmax)	-	25
product	0.14	0.14	text	lstm	64(Lstm, relu), 32 (Lstm, relu), 128(dense,relu),1022(dense,softmax)	-	25

Overall, under the given hyperparameter tuning the models that performed best where the ones that were dense and received the text (long text) feature as input. In addition, the poor performance of all the models regarding the prediction of the product label is mainly caused by the significant misrepresentation of a large number of classes in the dataset. For instance, there was only one training example for certain training classes not to mention that a large number of classes did not contain more than 10 training examples. To conclude, the models that were chosen for the competition were the dense models with input title.

The user name in Codalab is Alexandros_F and the score can be founded in the following link:

https://codalab.lisn.upsaclay.fr/competitions/19955#results