Overview

This project is email sentiment analysis through Convolutional Neural Network. We adapted a general-purpose email dataset from Kaggle by assigning sentiment to each email, then used the derived dataset to train a CNN model.

This CNN model can be deployed in production environment. Performing sentiment analysis on emails will allow companies to know if incoming messages are positive, negative, or neutral, and provides them with real-time insights to prioritize accordingly.

Another example implementation can be to enable product managers to understand customer emotions in their marketing campaigns. It is an important factor when it comes to product and brand recognition, customer loyalty, customer satisfaction, advertising and promotion's success, and product acceptance

Implementation

The project in implemented in two Jupyter Notebooks. Sentiment_CNN_v1.ipynb, email_data_cleaning.ipynb is where downloaded email csv file is processed and annotated for the sentiment analysis. In that notebook we first defined some utility functions to help with cleaning text data. Next the downloaded csv file is imported and explored to determine the original state the data is in.

We then filtered the dataset to retain only rows containing emails. The emails are then cleaned by removing leading and ending white spaces from the text, removing consecutive white spaces as well as removing unwanted characters from the emails.

Finally, the filtered and cleaned emails are passed through the "`vader_lexicon`" module in the nltk package to assign sentiment label for each email. The sentiment for an email is either "neu" for neutral, "neg" for negative or "pos" for positive. Email annotation is done by looping through emails. Each email is passed through a "`SentimentIntensityAnalyzer`" object to retrieve its polarity scores. The retrieved polarity values are compared against values we set in order to assign the label for the email. Our values set to compare against polarity values were assign after some trial and error. After several tests we settled on classifying any email whose negative polarity score is less than ".1" as negative, if the email's positive polarity value is less than '.15', it is assigned a positive sentiment label. If either polarity scores do not fall into the positive or negative categories, the email is classified as neutral. These values can be tweaked for sensitivity.

The email texts and assigned labels are then save in a new data frame that we exported as our training dataset.

The second notebook, '`Sentiment_CNN.ipynb`' [link to the uploaded notebook] is used to preprocess our dataset and train and export the CNN model. The Pytorch library is used.

The Sentiment_CNN notebook consists of four parts primarily. In the first part of the notebook, we import the labelled email dataset that was prepared in the '`email_data_cleaning`" notebook. In this section we also perform basic text cleaning and convert each email into a list of words.

In the second section we create a '`TextDataset`' class that is used to create word embedding from our lists of words previously created. The embedding is creating a vocabulary from our corpus of email. Values for padding, end of sentence characters and unknown words are added to the vocabular. The positions of the words in our vocabulary, using a dictionary, are used as embedding for the actual words. There are few more helper functions in the class. One function returns an email and its label when supplied an index. Another helper functions an email as LongTensor when provided an index, yet another returns the email label (0 for 'neg', 1 for 'pos' and 2 for 'neu') converted to LongTensor.

In the third section we create the Convolutional Neural Network. The CNN is a simple one initialized with an embedding layer created from our emails. In other methods in this section, we create a method to initialize the CNN model and another method to train it. There is also a utility function to specify the model's parameters. To complete this section, we train the model for 25 epochs, printing out the epoch, loss and training accuracy percentage for each epoch.

In the final section we evaluate the trained model on our test dataset. Trained for 25 epochs, the model was up to 75% accurate. The trained model is then exported as 'CNN.pt" and can be reused in various applications for sentiment classification. The model's accuracy can be improved by training it for more epochs

Usage

We were able to finish the model with good accuracy. The model was able to predict the sentiments of the emails as expected.

We did a local run of the software for demo purpose and placed it under the /Usage folder in github.

Team member contribution

- Analysis and research (16 hrs.) – Siva Prakasini Veera Sikku
- Gather and clean data (5 hrs) – Pradosk Kumar Subudhi
- Build machine learning model (25 hrs) – Aaron Dunor
- Train and evaluate model (3 hrs) – Aaron Dunor
- Team collaboration (15 hrs) – Aaron Dunor, Siva Prakasini Veera Sikku, Pradosk Kumar Subudhi
- Project documentation (4 hrs) – Pradosk Kumar Subudhi, Aaron Dunor, Siva Prakasini Veera Sikku
- Demo preparation and presentation(1hr) - Pradosk Kumar Subudhi, Aaron Dunor, Siva Prakasini Veera Sikku