Aspect-Based Sentiment Analysis

Team 42: MAGG

Manas Yendluri(2019113008)
Jai Ganesh(2019113023)
Aswin Jose(2019113016)
Ganesh CGS(2021713002)

Introduction:

The topic in the current discussion is Aspect Based Sentiment Analysis(ABSA). In this project, we tried to familiarize ourselves with ABSA by taking a further step in understanding by trying to implement two methods named Aspect Category Sentiment Analysis(ACSA) and Aspect Term Sentiment Analysis(ATSA).

- ACSA focuses on the category to which the aspect term belongs. For example, in the sentence “The sushi was bad.”, ACSA conveys that the category “food” was bad.
- On the other hand, ATSA predicts sentiment related to each and every term. So, in a case like “The sushi was bad but the rice was good”, ATSA can convey the sentiment associated with both sushi and rice individually and not as one broad category named “food”

We shall now take a brief look at the models used, comparisons, and more.

Workflow:

We initially began not by directly dealing with ABSA on the datasets used by the paper, rather we used a different dataset called the “IMDb dataset” for running and testing our basic models. Our workflow for achieving sentiment analysis was as follows

1. Pre-processing data and making it more usable by us.
2. Embedding these data using either glove vectors or sentence transformers or vectorizers.
3. Feeding this data into our models
4. Obtaining desired results

Pre-processing:

The datasets used by us in our codes are the ones used by the authors of the paper. The data was presented to us in .json format. We coded up a separate file just to handle the processing of .json into .csv while editing titles etc. These .csv files were the data files for further codes.

Embedding:

We used GloVe vectors as well as vectorizers or sentence transformers. The reason behind using three different embeddings was just to compare how each embedding affected a model and to try and see if we could get any useful information.
While using GloVe, we picked the 300d glove file to embed our data. Parallelly, we also used vectorizers, which convert words to vectors to see the difference in the efficiencies. For SVM, we ran the vectorizer as an embedder and we also used sentence transformers as the embedder in another run for SVM. Once this embedding was done, the outputs were directly fed into our models to obtain results.

Models:

We coded a handful of models starting with a very basic CNN model up to a Gated Convolutional network with Aspect Embedding(GCAE). A short description of these models is as follows

- Basic CNN: This was the very first model we coded. This model as the name suggests had nothing too extravagant related to it. It had a basic convolutional layer, whose output was fed into a max-pooling layer. The output of the max-pooling layer was sequentially passed through a ReLu activation layer followed by a Sigmoid activation layer.
- Basic LSTM: This model was a little simpler than our basic CNN model in the coding sense since it had 3 LSTM layers (from Keras) and the output from those was fed to a sigmoid activation layer.
- CNN and LSTM model: This is a little complicated model since it involved both CNN and LSTM in the models. It initially had a convolutional layer, followed by a max-pooling layer. This was then fed into an LSTM layer. This was then sent through a sigmoid activation layer, followed by a ReLu activation layer and another sigmoid activation layer.
- ACSA model: This model was our replication of the work done by the authors in the ACSA task. This model was rather complicated. It involved using embedded words and passing them through a convolution layer while parallelly passing embedded categories into a convolution layer followed by merging the outputs of these parallel layers. This output was then passed through a gated TanH-ReLu layer, followed by a max-pooling layer. The final output was passed through a sigmoid activation layer.
- ATSA model: This model too was equally complicated as the ACSA model. The difference was that both these embedded layers were passed through a convolutional layer and a max-pooling layer before being merged.
- SVM model: This model was coded to compare our models with a standard SVM model. This model involved running SVM with different kernels.

Note that all these models were run using the Adagrad optimizer, with a learning rate of 0.01 over 10 epochs.

Results and discussion:

Now we shall take a look at the results we obtained throughout the course of our tinkering.

- **IMDb dataset:**

  We ran our basic CNN model, LSTM, and LSTM+CNN model on the IMDb dataset and obtained the following results. (Adam optimizer was used for these models, as opposed to adagrad for the other ones)

Model	Performance on train data	Performance on test data
CNN	0.8999	0.8493
LSTM	0.6269	0.6291
CNN+LSTM	0.5008	0.4964

  We observe that CNN has a better accuracy by far over the other models. This is attributed to the fact that CNN performs extremely well in cases where a substantial feature needs to be detected, as opposed to LSTM, which is good for data that have sequential aspects. Sentiment analysis falls in the feature detection category, hence making CNN superior over LSTM.

- **SemEval dataset:**

  We ran our basic CNN, LSTM, and LSTM+CNN models on a specific category yet again on the SemEval dataset and obtained the following results

Model	Performance on train data	Performance on test data
CNN	0.6495	0.6482
LSTM	0.5782	0.6351
CNN+LSTM	0.5782	0.6351

  We observe that CNN's superiority in accuracy was reduced here. This is due to the fact that the SemEval dataset is much smaller than the IMDb dataset, hence hindering the models from learning better.

- **SVM**

  The SVM model was run on the SemEval dataset and the following results were seen.

Embedding	Kernel	Test accuracy
Vectorizer	Linear	0.8361
Vectorizer	Sigmoid	0.8361
Vectorizer	RBF	0.8407
Vectorizer	Poly	0.7815
Sentence Transformers	Linear	0.8828
Sentence Transformers	Sigmoid	0.8896
Sentence Transformers	RBF	0.9067
Sentence Transformers	Poly	0.9135

  A very important point of notice is that, for each kernel, the sentence transform (SentenceBERT) embedding performed better than the vectorizer embedding. This is because the vectorizer is far less complicated than the SentenceBERT, which is based on an advanced framework used in NLP, clearly making it more suitable for sentiment analysis.

- **ACSA**

  The ACSA model gave us the following results

	Loss	Accuracy	F1 score	Precision
Test data	0.3304	0.8577	0.9024	0.8718
Train data	0.2111	0.9210	0.9458	0.9311

  The loss reported here was the binary cross-entropy loss, as it is the best loss function for binary classification problems. Adagrad optimizer was used here since the authors of the paper did too. We observe an 85.77% accuracy in test data by ACSA model, which is in the perfect margin obtained by the authors of this paper.

- **ATSA**

  The ATSA model gave us the following results

	Loss	Accuracy	F1 score	Precision
Test data	0.4991	0.7751	0.8206	0.7658
Train data	0.3016	0.8869	0.9921	0.8962

  Here we observe that ATSA is clearly underperforming ACSA in terms of both loss and accuracy. We see a 77.51% accuracy in the ATSA model again which is in excellent agreement with the accuracies obtained by the authors.

  But why does ACSA perform better than ATSA? We believe that is because ACSA predicts a broader range of things and not something very specific like ATSA does. This loss in specificality is what leads to an increase in the accuracy. We also notice how ACSA performs slightly better than SVM but ATSA underperforms SVM. The performance of SVM largely depends on the availability of features for the case of ACSA. Also, the presence of gated units brings more accuracy to ACSA over SVM.

Concluding remarks:

The usage of gated units and convolutional layers in ACSA and ATSA, make them extremely efficient for parallel computing. The difference in times between ACSA/ATSA and the other models was anywhere between 1 to 10 times the time taken by ACSA/ATSA. Although not the best in terms of accuracy when compared to other salient methods in ABSA, this method surely is one that stands out. Overall, the methods used here have the potential for future use owing to their promising performances.