SMAI Final Report | Buzz

Team Buzz

- Alapan Sau (2019101081)

- Pavani Babburi (2019101033)

- Shri Vidhatri M M (2019113006)

- Poojitha Mittapalli (2019102018)

Problem Statement

Mining the opinion of the reviews has become an important problem in recent years due to the increasing popularity of online retailers, review blogs, etc. This helps us gauge the user sentiment towards a particular product or service that is of use to many entities, including prospective customers, business owners and service providers.

For sentiment analysis of reviews, each sentence of the review are usually classified independently, although they form a part of the review's argumentative structure. A the sentence in a review is rarely an individual entity, and more often than not, is a part of a collection of sentences which build and elaborate on each other, and together form the case either for or against the product.

The proposed method uses a hierarchical bidirectional LSTM for the task of aspect-based sentiment analysis of reviews, which models the interdependencies of sentences in a review.

Proposed Architecture

The architecture proposed in the paper for aspect-based sentiment analysis is as follows:

1. Word Embedding Layer

2. Aspect Embedding Layer

3. Sentence-Level Bi LSTM

4. Review-level Bi LSTM

5. A dense layer and a SoftMax layer
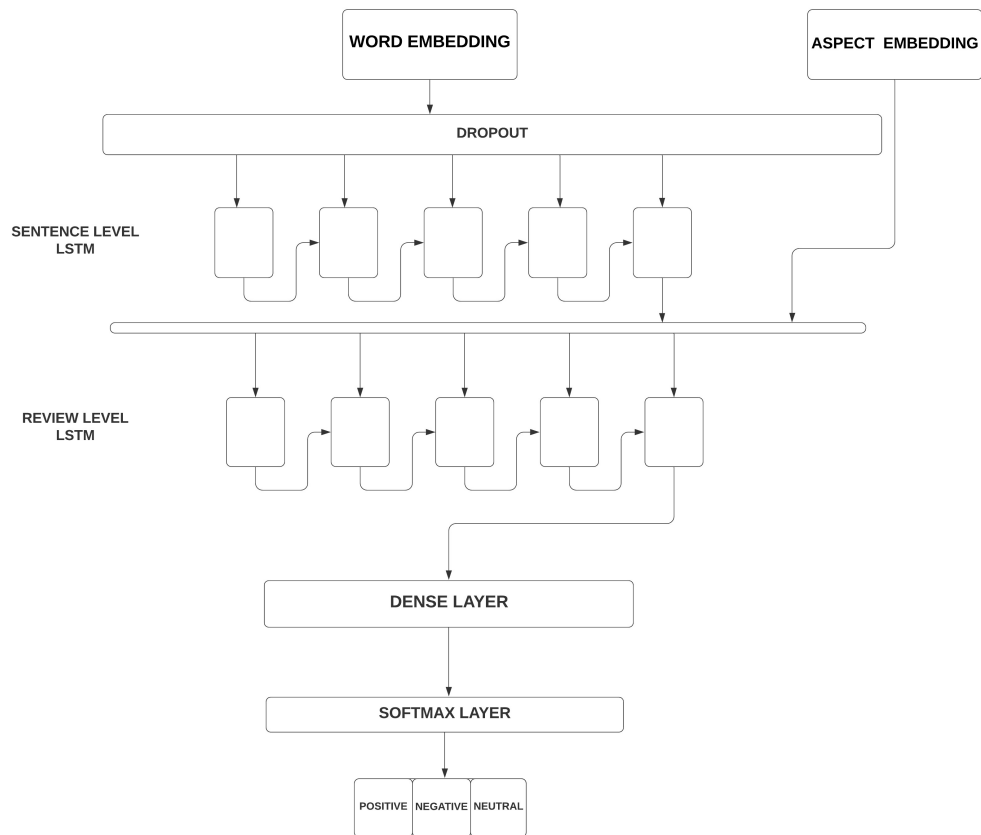
The various layers are described below, along with essential code snippets:

Word Embedding Layer

The paper utilizes two-word embedding layers:

GloVe pre-trained Embeddings

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. We use 300-D word embeddings trained on the Wikipedia Dataset. As we use pre-trained embeddings, no further training is needed in the embedding layer.

```python
class Embed(nn.Module):

    def __init__(self, weights_matrix):
        super(Embed, self).__init__()

        self.num_embeddings, self.embedding_dim = weights_matrix.shape
        self.emb_layer = nn.Embedding.from_pretrained(torch.FloatTensor(weights_matrix))
        self.emb_layer.weight.requires_grad = False

    def forward(self,x):
        out = self.emb_layer(x)
        return out
```

Random word Embeddings

The word embeddings are initialized randomly and trained along with the other parameters of the network.

```python
class WordEmbedding(nn.Module):

    def __init__(self, num_embeddings, embedding_dim):
        super(WordEmbedding, self).__init__()
        self.num_embeddings = num_embeddings
        self.embedding_dim = embedding_dim
        self.emb_layer = nn.Embedding(num_embeddings, embedding_dim)
        self.emb_layer.weight.requires_grad = True
```

A dropout layer with a probability of 0.5 is used after the embedding layer.

Aspect Embedding Layer

Embeddings for the aspects are randomly initialized 15-D vectors, which are trained in the network.

Aspects consist of an entity and an attribute, and the aspect of the review is obtained by averaging the vector of the entity and the attribute

```
class AspectEmbeddings(nn.Module):

    def __init__(self, num_embeddings, embedding_dim):
        super(AspectEmbeddings, self).__init__()
        self.num_embeddings = num_embeddings
        self.embedding_dim = embedding_dim
        self.emb_layer = nn.Embedding(num_embeddings, embedding_dim)
        self.emb_layer.weight.requires_grad = True
```

Sentence-level Bi LSTM

Sentence level Bi LSTM deals with the individual words of the sentence. Each sentence returns two 200-D vectors, which are fed into the Review-level Bi LSTM

```
class WordBiLSTM(nn.Module):

    def __init__(self, input_dim, output_dim, num_layers=1, dropout=0, rnn_type=nn.LSTM):
        super(WordBiLSTM, self).__init__()

        self.input_dim = input_dim
        self.num_layers = num_layers
        self.dropout = dropout
        self.output_dim = output_dim

        # LSTM layer
        self.lstm = rnn_type(self.input_dim, self.output_dim, self.num_layers,
        batch_first=True, bidirectional=True, dropout = self.dropout)
```

Review-level Bi LSTM

Review level Bi LSTM operates on the level of reviews. Each sentence is concatenated with its aspect and fed into the Review-level Bi LSTM

```python
class SentBiLSTM(nn.Module):

    def __init__(self, input_dim, output_dim, num_layers, dropout, rnn_type=nn.LSTM):
        super(SentBiLSTM, self).__init__()

        self.input_dim = input_dim
        self.output_dim = output_dim
        self.num_layers = num_layers
        self.dropout = dropout

        self.lstm = rnn_type(self.input_dim, self.output_dim, self.num_layers,
        batch_first=True, bidirectional=True, dropout = self.dropout)
```

Output Layer

The output from bi-directional LSTM passed to a dense layer which converts the data to vectors,
which are then passed to a SoftMax layer to generate the probabilities across the three labels (positive,
negative, and neutral).

H-LSTM

All the classes mentioned above are in the pipeline by the H-LSTM class, which is the final model.

```python
class HLSTM(nn.Module):

    def __init__(self, embedding_type = "glove"):
        super(HLSTM, self).__init__()

        if (embedding_type == "glove"):
            self.embed = Embed(weights_matrix)
        else:
            print("Initialising random word embeddings\n")
            self.embed = WordEmbedding(num_target_vocab, 300)

        self.aspectEmbed = AspectEmbeddings(num_aspect_vocab, 15)
        self.dropout = nn.Dropout(0.5)
        self.wordLSTM = WordBiLSTM(input_dim = 300, output_dim=200,
                        num_layers=1, dropout=0, rnn_type=nn.LSTM)
        self.sentLSTM = SentBiLSTM(input_dim = 415, output_dim=200,
```

```
                          num_layers=1, dropout=0, rnn_type=nn.LSTM)
        self.linearLayer = nn.Linear(400, 3)
        self.softMaxLayer = nn.Softmax(1)
```

Data Collection and Preprocessing

The data for the project mentioned above is related to the SemEval2016 Task5 dataset. The Website corresponding to this task has removed the dataset, and due to copyright issues, accurate training data was not readily available. Hence, we had to resort to finding sample datasets on the Internet.

After searching through the Internet and parsing the dataset, we were able to collect the following datasets:

1. English Restaurants Dataset 1 - 413 Reviews consisting of approximately 1000 sentences (400 Training and 13 Test)

2. English Restaurants Dataset 2 - 521 Reviews consisting of ~20 sentences each (500 Training and 21 Test)

3. English Laptop Dataset - 357 Reviews consisting of ~20 sentences each (350 Training and 7 Test)

The data is padded accordingly to account for variable lengths.

Considering the little data available, we weren't able to train the network well enough to replicate the exact results of the paper. However, we were able to obtain some interesting results.

Training Details

The details of the training are as follows:

- Dropout after embedding layer with $p = 0.5$

- A gradient clipping norm of 5 after LSTM

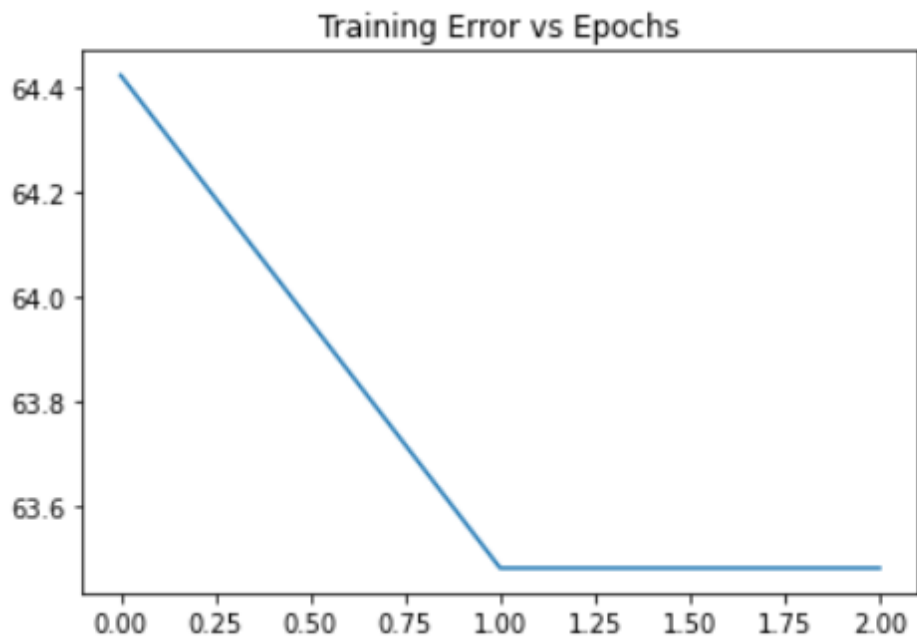- The loss function used is Cross Entropy Loss

- Adam update rule is used with stochastic gradient descent, with a mini-batch size of 10
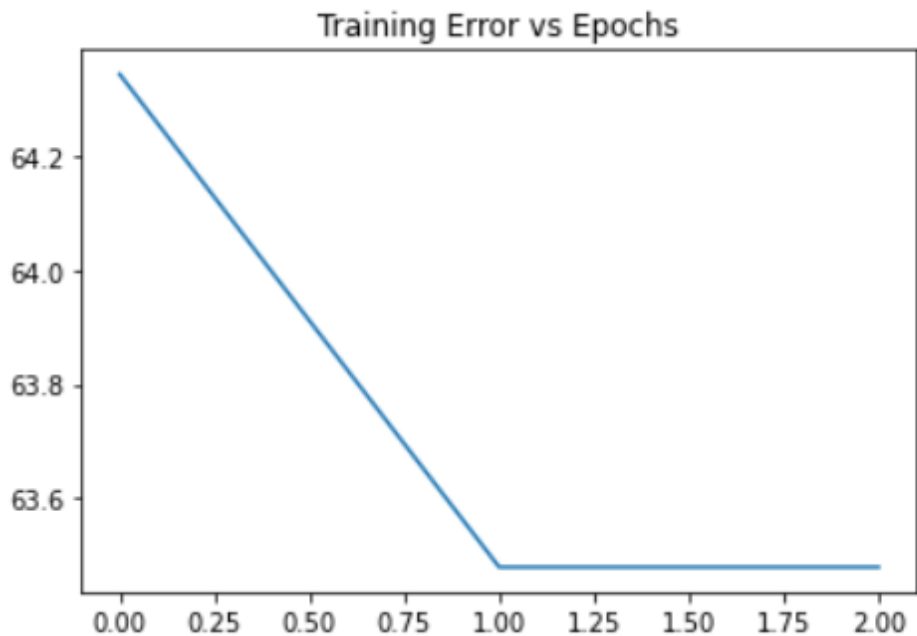
Observations

The results observed for each dataset are as follows:

Restaurants Dataset 1 (English)

Due to the limited data, the model trained for very few epochs, whose errors are plotted below:

Training Error vs Epochs for GloVe embeddings

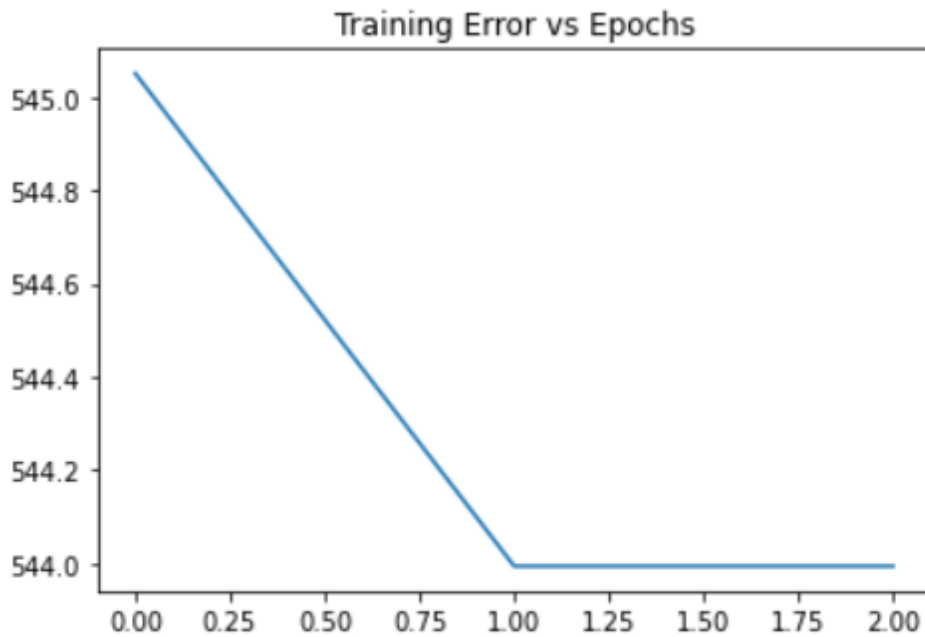**Training Error vs Epochs for random embeddings**

The test accuracy for this dataset is 82.7% for both embeddings. While this seems impressive, it must be noted that the test sample is tiny, and hence, isn't conclusive enough to comment on the general accuracy of the model.

Interestingly, both GloVe embedding layers and random embedding layers accuracy are pretty similar, and the training error is less for random embeddings than GloVe embeddings on some rare occurrences. This is probably because the model hardly undergoes any training, and hence there isn't any significant improvement from the initially assigned values. This high accuracy is surprising considering the little training, but it's probably the data itself rather than the model performance.
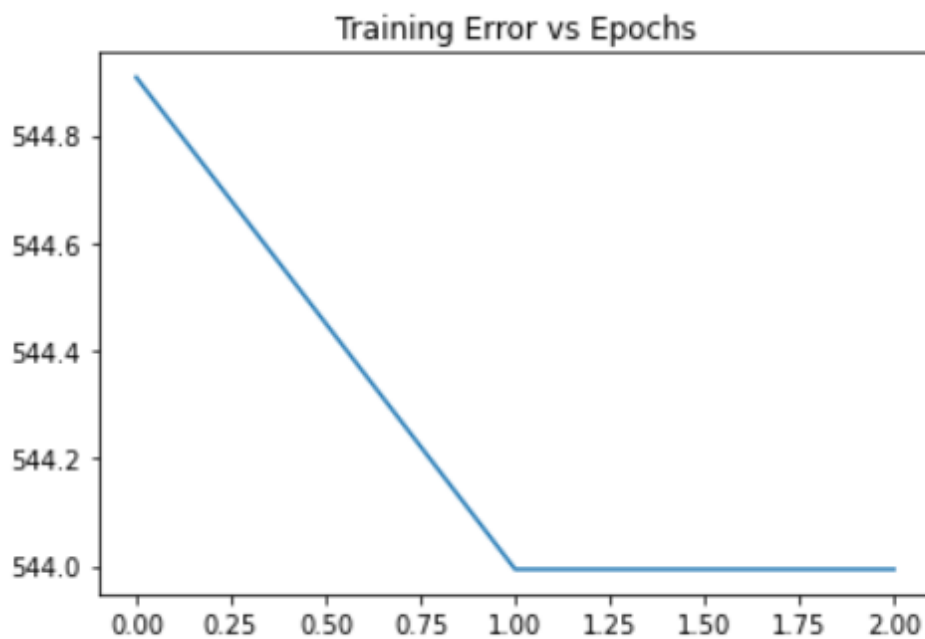
Restaurants Dataset 2 (English)

The plots for the second dataset is as follows:

Training Error vs Epochs

Training Error vs Epochs for GloVe embeddings
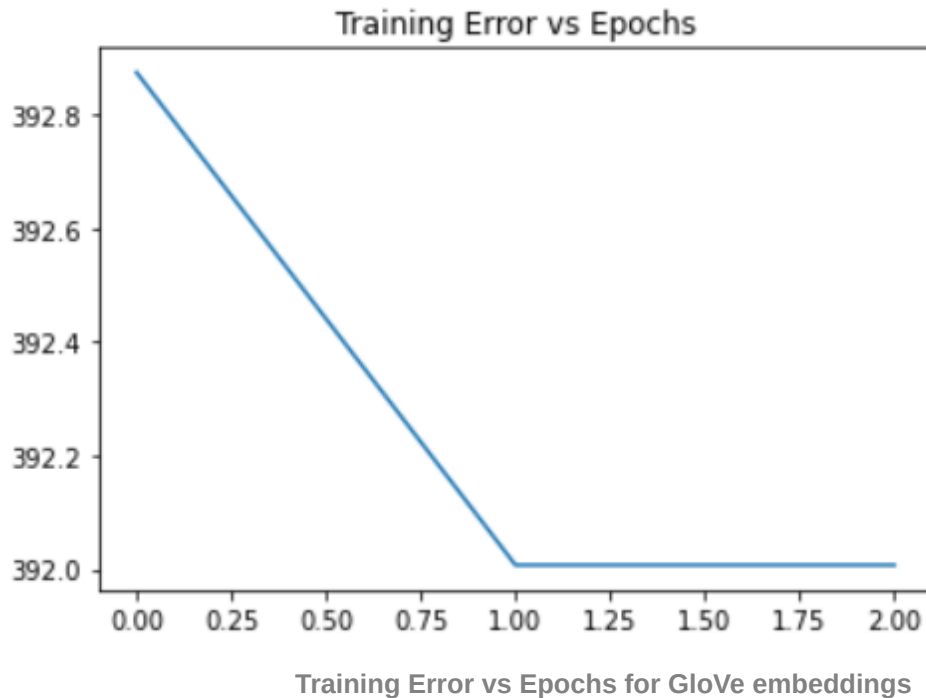
Training Error vs Epochs

Training Error vs Epochs for random embeddings

The test accuracy is again the same for both the embeddings (59.9%), but this is very low compared to the previous one.

Laptop Dataset(English)

The error plot for the English Laptop Dataset is as follows:

Training Error vs Epochs for GloVe embeddings

This dataset gives the least accuracy of 16% among all the datasets. It's not clear whether domain difference plays a significant role in the amount of data required, or it is just an issue of similarity between the available datasets.

How much is data significant?

Data drives deep learning models. This is especially true for LSTMs and hierarchical models. Considering that our model combines both, low data hardly give any meaningful results. It primarily provides a general idea of whether the code is working. However, the model efficiency cannot be analyzed with such little data.

We have observed the model trains for very few epochs (2-3). Also, adding layers in the earlier parts of the layer (For example, Dropout after embedding) hardly changes the final output of the data, signifying that the initial layers are not getting trained effectively. While we couldn't pinpoint an exact reason for this, the lack of data leading to minimal gradients and updates seems a good guess.

Can we do better with the available data?

A possible way to improve the training, especially with such low data, would be to use transfer learning. We train the model initially with some well-known NLP tasks with enough data and then train the final few layers with the data for the specific task. This can also be extended for this particular task such that we train the initial layers with all the datasets combined and then train the last layers with the specific dataset. However, we couldn't explore this further in the interest of time.

Work Distribution

Aa Name	☰ Work assigned
Shri Vidhatri M M	Dataset preparation
Shri Vidhatri M M	Data preprocessing
Pavani Babburi, Alapan Sau	GloVe Embeddings preperation
Pavani Babburi, Alapan Sau	Embedding Layers
Pavani Babburi, Alapan Sau	Sentence level and review level Bi directional LSTM
Pavani Babburi, Alapan Sau	Hierarchical Bi-Directional LSTM
Pavani Babburi, Alapan Sau	Training the model

Deliverables

The entire code and datasets used for this project can be found at:

https://github.com/alapan-sau/SMAI-Project

References

1. https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html

2. https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

3. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

4. https://github.com/jiangqn/Aspect-Based-Sentiment-Analysis

5. https://github.com/cedias/Hierarchical-Sentiment

6. SemEval-2016 Task 5: Aspect Based Sentiment Analysis https://aclanthology.org/S16-1002.pdf

7. Pal, Subarno & Ghosh, Soumadip & Nag, Amitava. (2018). Sentiment Analysis in the Light of LSTM Recurrent Neural Networks. International Journal of Synthetic Emotions. 9. 33-39. 10.4018/IJSE.2018010103.

8. Long, Fei & Zhou, Kai & Ou, Weihua. (2019). Sentiment Analysis of Text-Based on Bidirectional LSTM With Multi-Head Attention. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2942614.

9. https://github.com/howardhsu/ABSA_preprocessing/tree/master/dataset/SemEval/16

10. https://alt.qcri.org/semeval2016/task5/data/uploads/trial-data/english-trial/restaurants_trial_english_sl.xml

11. https://alt.qcri.org/semeval2016/task5/data/uploads/trial-data/english-trial/restaurants_trial_english_tl.xml

12. https://alt.qcri.org/semeval2016/task5/index.php?id=data-and-tools