Aspect-based sentiment analysis

Parsa Mazaheri
UC, Santa Cruz pmazaher@ucsc.edu

Abstract

1 Introduction

Aspect-based sentiment analysis is a sub-field of NLP and the task of identifying fine-grained opinion polarity towards a specific aspect associated with a given target.

2 Problem Description

Sentiment analysis is a natural language processing (NLP) technique used to determine whether data is positive, negative or neutral. By contrast, with aspect based sentiment analysis (ABSA), the goal is to identify the aspects of given target entities and the sentiment expressed towards each aspect.
Since here we're using labeled dataset to train the model and then use it for prediction, we're using **Supervised Learning** approach. supervised learning is a machine learning approach where we have a dataset of labeled data and we train a model to predict the label of new data.

3 Aspect-Based

Aspect-based sentiment analysis is a topic-based sentiment analysis and is more productive since it focuses on aspects of the sentiment rather than just focusing on the whole score. ABSA has different kinds can be different by getting the polarity of a term or getting the context and terms from the sentiment.

3.1 Subtask 1: Aspect term extraction

Given a set of sentences with predefined entities, identify the aspect terms present in the sentence and return a list containing all the distinct aspect terms. An aspect term names a particular aspect of the target entity.

3.2 Subtask 2: Aspect term polarity

For a given set of aspect terms within a sentence, determine whether the polarity of each aspect term is positive, negative, neutral or conflict (both positive and negative). For example: for the given sentiment "I hated their fajitas, but their salads were great", we should get the following result for the aspects in the sentiment.
`{fajitas: negative, salads: positive}`

3.3 Subtask 3: Aspect category detection

Given a predefined set of aspect categories, identify the aspect categories discussed in a given sentence. For example, given the set of aspect categories food, service, price, ambience: "The restaurant was expensive, but the menu was great" the model should give `{price, food}` which are the aspects in the sentiment.

3.4 Subtask 4: Aspect category polarity

Given a set of pre-identified aspect categories, determine the polarity of each aspect category. For example: for the given "The restaurant was expensive, but the menu was great" the model should give price: negative, food: positive.

4 Dataset

For dataset we're using the Semeval 2014 dataset (Pontiki et al., 2014) which includes reviews for laptops and restaurants. This dataset was designed for the task of sentiment analysis in different aspects including Price, Food, Service, Ambience, and Anecdotes. We've separated the data by aspect for better working on each of them but for the testing, we've used the whole data all at once.
The dataset consist of 3841 data which 3041 are in the train data and 800 in the test data. Also Aspects distribution per sentiment class is shown in the below.

=pt

Aspect	Positive	Negative	Conflict	Neutral	Total
Food	1169	278	82	121	1650
Price	230	143	20	11	404
Service	425	281	40	23	769
Ambience	339	119	60	31	549
Anecdotes/Misc.	673	240	45	408	1366
Total	2836	1061	247	594	4738

Figure 1: Aspects distribution per sentiment class

5 Preprocessing

Data preprocessing is a step in data analysis that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning models. Good, preprocessed data is more important than the most powerful algorithms. By doing a good preprocessing on the data, we can save a lot on optimizing the model, hyperparamters, and trying to generalize the problem because of the diversity of the data. The preprocessing consists of different actions on the data which has been explained briefly in below.

5.1 Tokenization

In this step, we remove the punctuation, removing the stop words, and other tasks which here is done using Bert Tokenizer which is a pre-trained BERT model and because of this we don't need to do these things which was mentioned ourselves.

6 Hyperparameters

One of the key challenges of every deep learning task is to determine the best hyperparameters. The hyperparameters are the parameters that are used to train the model. The hyperparameters are not learned by the model and therefore aren't updated during the training process. The hyperparameters that I've used for this task are shown in the following table.

Hyperparameter	Value
Learning Rate	2e-5
Batch Size	24
Hidden Size	768
Number of Epochs	4
Number of Layers	12
Number of Attention Layers	24
Dropout	0.1

Table 1: Hyperparameters

7 Model Architecture

For this project we have used BERT [1] model which makes use of Transformer, an attention (Vaswani et al., 2017) mechanism that learns contextual relations between words (or sub-words) in a text.

7.1 BERT

BERT's (Devlin et al., 2018) key innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous model architectures which looked at a text sequence to sequence and either from left to right or combined left-to-right and right-to-left training. Instead the model is bidirectionally trained and can have a deeper sense of language context and flow than single-direction language models. BERT uses a technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

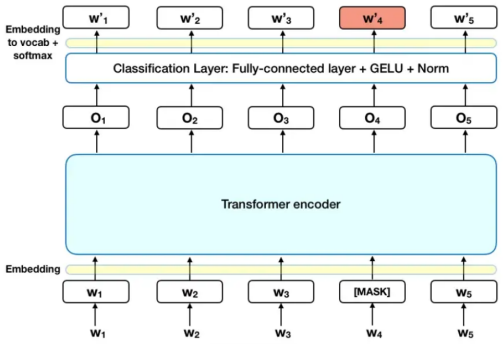

Figure 2: BERT architecture

Before feeding word sequences into BERT, 15 percent of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.

7.2 Fine-tuning Bert For our Problem

While BERT is a state-of-the-art model for NLP tasks, it can have multiple functionalities based on our use-case by adding a small layer to the core model. Classification tasks such as sentiment analysis (as well as aspect-based) are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.

[1] Bidirectional Encoder Representations from Transformers

=pt

Learning rate	Attention layers	F1 Score
lr=0.001	layers=24	0.7091
lr=0.001	layers=12	0.6743
lr=2e-5	layers=24	0.8208
lr=2e-5	layers=12	0.7583

Table 2: Learning rate and number of Attention layers influence on F1 score

In the fine-tuning training, most hyper-parameters stay the same as in BERT training. The paper itself gives specific guidance (Section 3.5) on the hyper-parameters that require tuning which in our model we're going to use the original ones.

8 Experiments

Since for this research project, as mentioned in dataset section, we're going to use semeval 2014 dataset (Pontiki et al., 2014) which consists of these 5 aspects including price, anecdotes, food, ambience, and service.

Also, since for this project, we're using HugginFace, we can have better safely work with different hyperparameters for the problem.

8.1 Hyperparameters

8.1.1 Learning rate

For learning rate, a too high learning rate will make the learning jump over minima but a too low learning rate will either take too long to converge or get stuck in an undesirable local minimum. For bert we usually use learning-rate in the order of 1e-5 - 2e-6. I've tested learning-rate from `0.001` to `2e-5` and the best value was achieved using `2e-5`. For the 0.001 learning-rate, the F1 Score of the result decreased drastically which a sample comparison is shown for them in table 2.

8.1.2 Batch Size and Weight Decay

For batch size, 32 here is pretty standard and based on the size of our dataset, it's good choice to use and we stick with it here as well.
Weight decay is a regularization technique that is used in machine learning to reduce the complexity of a model and prevent overfitting. Here we use weight decay as well for overfitting with `0.01` value.

8.1.3 Epochs

For epochs, again since we're using a pre-trained BERT model and just tuning it dfor our task, we don't need to use large EPOCHS and can use 4 EPOCHS for each of the aspects. The Model runs on each aspect and evaluates itself on that so that the model be able to do well on all aspects. Also, we saved the model at each checkpoint to have the saved weights for our model.

8.2 Number of layers

For the BERT model, I tried different combination of attention layers and hidden layers to find the best combination. The more attention layers can help with better getting the context. In my experiments, increasing the number of attention layers to double (12 to 24) **reduced the precision but increased recall and overall F1 score** which is shown in Table 3.

8.3 Dropout

Dropout (Srivastava et al., 2014) is a technique for regularization. Dropout is a technique where randomly selected neurons are ignored during training. by using dropout in this problem, I didn't encountered much difference, since we're dealing with a small number of data and don't have very deep neural networks, that help with data loss.

9 Result

Evaluation is scored based on mean F-1 score. This is a common evaluation metric across NLP, because it weights both precision and recall.

$$F1 = \frac{2.P.R}{P+R}$$

in which Precision is a ratio of True Positives (TP) to total positives guessed (TP + FP). Recall is the ratio of True Positives to actual positives (TP + FN).

$$P = \frac{TP}{TP+FP}, P = \frac{TP}{TP+FN}$$

By using F1-Score as the evaluation metric, the model is able to achieve a score of 0.9789. This is a good score for a model that was trained on a small dataset without very intensive hyperparameter tuning and model optimization.

By using F1-Score as the evaluation metric, the model is able to achieve a score of **0.9789**. This is a good score for a model that was trained on a small dataset without very intensive hyperparameter tuning and model optimization.

The result of using dropout on the model is shown on table 3. As mentioned in section 8.3,

=pt

Attention Layers	Precision	Recall	F1 Score
12 Layers	0.9541	0.6292	0.7583
24 Layers	0.9396	0.7287	0.8208

Table 3: Final Result for the model

Dropout Rate	F1 Score
0.1	0.8208
0.3	0.7993
0.5	0.7501

Table 4: Dropout influence on the score

since here we don't have a deep network and a large dataset, dropout doesn't help very much and as seen in in the Table 3, with dropout `value=0.1`, the accuracy of the result improves but if we increase dropout rate to 0.5, the accuracy drops very quickly and noticeable.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

=pt