Syntax and Semantics ensemble model for Aspect-based Sentiment Analysis

EE8607

Hoang Nguyen
Department of Computer Engineering, Ryerson University
December 4, 2021
hoang.cam.nguyen@ryerson.ca

Abstract

Aspect-based sentiment analysis is a fine-grained sentiment analysis which aims to provide the sentiment polarity towards a specific term in a sentence. Many studies focus on employing attention mechanism and positional embedding to implicitly measure the semantic relationship between the aspect terms and sentences. However, these models still neglect the complex semantic and syntactic structure of natural language. In this study, we present a framework which encodes these two information by base learners, a transformers-based module to extract the semantic and a graph neural network-based module to capture the syntactic representation. Extensive experiments on SemEval 2014 dataset demonstrate a significant and consistent improvement in our proposed model quality.

1 Introduction

Aspect-based sentiment classification is a long-standing and popular task in the natural language processing field [1, 2, 3, 4]. This task is usually performed on online affective texts, namely product reviews. The purpose of the task is to decide the sentiment polarities of aspect terms in a sentence. Assigning the whole sentence's polarity to every aspect is infeasible since some aspects may express different sentiments and viewpoints. That is the reason why aspect-based sentiment analysis enables us to offer deeper insights into texts or online reviews.

Some early works apply handcrafted features to train a statistical classifier for aspect-based sentiment analysis [5, 6], or some depend on handcrafted syntactic rules [7, 8]). Nonetheless, those approaches requires labor-intensive annotation process, which is alleviated by modern deep-learning architectures. For example, Convolutional Neural Architectures and Recurrent Neural Architectures have already been adopted to aspect-level sentiment analysis [9, 10]. Recently, as attention-based models have been emerging strongly, they are used to exploring the potential semantic connections between contextual words and given aspects [11, 12, 13, 14, 3]. In [15], the authors propose to use the attention mechanism in both sentence level and aspect term level to enhance the sentence embedding. Positional embedding is also introduced in [4, 15] in order to capture the importance of the words based on their relative position to the aspect term. Although these approaches show great results, they fall short on modelling the syntactical dependencies between words in the sentence, which better suits to show the interaction between them compared to the general positional embedding.

With the advance in Transformers, there are many research studies focus on exploiting Transformers-based models to extract the semantics of the sentence while doing the embedding [16]. Recent works such as [17, 18] also take advantages of syntactical information among contextual words by constructing relation graph from dependency parser and then encoding the graph structure with Graph Neural Network. However, these works either apply a pooling layer on the sentence-level to generate the representation of sentence or use pooling layer to encoder nodes representation to a unified graph representation. Besides, the word representation in the syntactical graph is usually initialized by a pretrained word embedding and it is different from BERT/Transformers due to the mismatching of

the output between the tokenizers in Transformers and dependency parser. Meanwhile, BERT uses subword tokenization algorithm, which may give more structural information rather the word-level tokenizer in dependency parser.

In this work, we aim to investigate the syntax information while aligning the word tokenizer between the BERT-based model and the dependency parser. In addition, we employ attention mechanism to encode the graph structure allowing us to better capture the graph embedding rather than a simple pooling layer. The output from BERT-based model (semantics) and graph-based model (syntax) are then fused to produce the final embedding. Finally, we examine the benefits of ensemble model by applying a classifier model on the predictions of base models. Extensive experiments are conducted on the SemEval 2014 dataset to show the effectiveness of our models along witht the ensemble learning.

2 Methodology

Figure 1 shows the overall architecture of the proposed model. There are two main components in our proposed model, one for extracting semantics and the other for syntax.

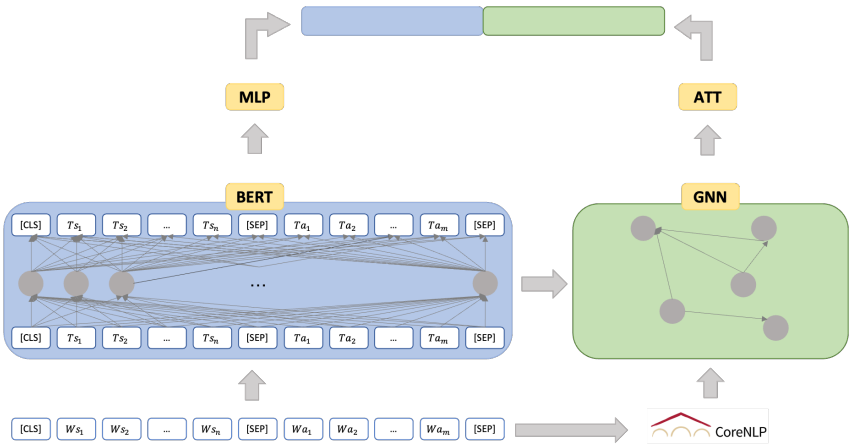

Figure 1: Overview of our proposed framework which consists of two submodules, the one on the left is for semantic information (*SEM*) and the one on the right is for syntactical information (*SYN*). *SEM* is a standard BERT-based model. *SYN* is graph neural-based model built from the aligned tokens and relation dependencies.

2.1 Word Tokenizer and Aspect-Aware Encoder

In order to tokenize and encode the text sentences with the existence of the aspect term, we adopted a standard pre-trained BERT model. Given the input sentence with m word tokens $S = w_{s1}, w_{s2}, ..., w_{Sm}$ and the aspect term with n tokens $A = w_{a1}, w_{a2}, ..., w_{an}$, we concatenated the sentence and the aspect term into a text sequence with the format $[[CLS], w_{s1}, w_{s2}, ..., w_{Sm}, [SEP], w_{a1}, w_{a2}, ..., w_{an}, [SEP]]$ and feed them to BERT as the input. As the result, BERT produces the sequence of tokens $T = t_{s1}, t_{s2}, ..., t_{sm}$ for the input text and their corresponding feature vectors. It should be noted that the length of the output tokens from BERT may be different from the original tokens since BERT splits a word which is not in its vocabulary into subwords and they are denoted with the prefix *"##"*. These feature vectors are than both used as the input of two submodules in the proposed model.

2.2 Graph Attention Network

A dependency tree can represent the syntax of a sentence, to be more specific, it expresses the syntactical dependencies between a word and its neighbor words. Figure 2 gives an example of the output from the dependency parser. To build the syntax graph for each sentence, we first adopt Stanford CoreNLP [19] to parse the input text and perform a token alignment between BERT and

CoreNLP tokenizer so that if a word is splitted by BERT, then a token in dependency parser is also splitted. Take the input text *a new Transformers* as an example. In dependency parser, the tree are built from the tokens a, new and Transformers with the indexes 0, 1, 2 respectively. Meanwhile, BERT tokenizes the text into a, new, Transform and ##ers with indexes 0 to 3. Hence, we will align the token in CoreNLP to match with BERT tokens.

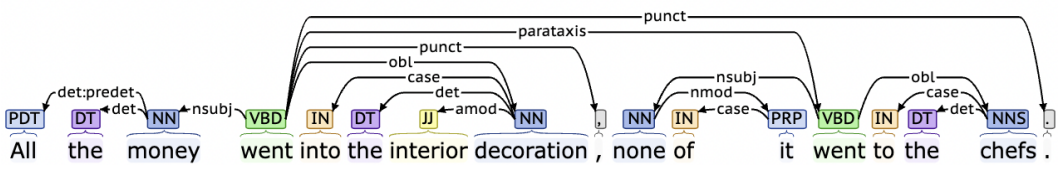

Figure 2: An example of dependency tree generated by Stanford CoreNLP [1].

After having the aligned tokens and the syntactic relationships, we build the graph to represent the input sentence in such the way that the nodes are the tokens (each subword is a token which means a word may contains multiple tokens) and two nodes are connected if there is a syntactic relationship between them. In the case of subwords, the syntactic edge only connects the node of the original word, and there are additional edges connecting the original word to its subwords. By this way, we can enable the embedding vector of the subwords produced by BERT to involve in the graph-based model training and make used of the semantic information as well.

Once having the graph built, we exploit Graph Attention Networks (GATs) to further learn the word embedding in the sentence by aggregating the embedding of neighbor nodes with multi-head attention.

$$h_i^{(l)} = \|_{k=1}^{K} \sigma\Big(\sum_{j \in \mathcal{N}_\rangle} \alpha_{ij}^{lk} W^k h_j^{(l-1)}\Big) \tag{1}$$

where $\|$ is the concatenation, α_{ij}^{lk} is the attention weights by k attention, and $\mathcal{N}_\rangle$ are the neighbor nodes of i.

Then, we adopt attention mechanism at the graph level to learn the node importance and finally, the graph embedding is computed by taking the weighted average of the nodes' embedding. The computations are shown in Equation 2 and 3.

$$c_i = \sigma(a_{intra}^T W_{intra} h_i) \tag{2}$$

where a_{intra} and W_{intra} are the learnable vector and matrix shared between nodes in the graph.

$$z = \sum c_i hi \tag{3}$$

2.3 Ensemble Learning

In this study, we adopt a supervised learning as a meta-learning strategy to combine the base model scores. Ensemble which is also known as super learner usually shows better results compared to the base learners if the base learners are diversified [20]. We leverage a supervised classifier, namely XGBoost [21], to input the predictions of three models: (i) **SEM**atic: we apply a softmax function on the contextual representation of pooling layer after BERT to get the prediction, (ii) **SYN**tax: a softmax function is apply on the unified graph embedding, and (iii) **S**emantic**S**yntax: a softmax function is applied on the concatenation vector of (i) and (ii); then train to produce the final ensemble predictions **S**emantics**S**yntax**E**nsemble.

2.4 Loss function

Finally, the standard cross-entropy loss is used as our objective function:

$$\mathcal{L}_{Classify} = -\sum_{i=1}^{D}\sum_{j=1}^{P} y_i^j log(\hat{y}_i^j) \tag{4}$$

where D is the number of sentence-aspect pairs, P is the number of polarities, and $\hat{y}_i = softmax(h_i)$ is the predicted polarity of each aspect term in its corresponding sentence.

3 Experiment

3.1 Implementation

All the experiments are conducted by using Pytorch [22], Pytorch Lightning [23] and Pytorch Geometric [24]. We would like to describe some main classes and functions.

Main directory:

```
.
├── src
│   ├── attention.py
│   ├── dataset.py
│   ├── inference.py
│   ├── main.py
│   ├── model.py
│   ├── parser.py
│   ├── tuner.py
│   └── utils
└── experiment
    └── 4_ensemble.ipynb
```

The main modules are in the following files.

- `src/parser.py`: This file consists some functions for preprocessing dataset. Two main functions are `parseXML` (to convert XML input file to dictionary) and `build_dep_parse_tree` (to build graph with the format of edge indexes from dependency parser)
- `src/dataset.py`: This file includes `Dataset` class and `DataModule` class. `Dataset` class is a Pytorch class with some extensions to encode the text sentence and build the adjacency matrix from edge indexes. `DataModule` is inherited from `pl.LightningDataModule` which is designed to load data for training, validation and testing. In this class, there is `collate_fn` function to customize the data loaded per batch since there are different types of data in `Dataset` such as text, tensor or vector and we also do batching and masking for graph data.
- `src/model.py`: This file includes model classes in this project. There are three classes: `SentimentClassifier` (SEM), `SynSentimentClassifier` (SYN) and `SynSemSentimentClassifier` (SS). All classes are inherited from

`pl.LightningModule` which the structure is in `forward` function, train in `training_step` in and validation in `validation_epoch_end`.

- `src/main.py`: main function to run training and testing. The training process including early stopping or checkpoint are defined in the `Trainer`
- `src/tuner.py`: This file is for hyperparameters tuning with *Ray Tune*. The search space of hyparameters is defined by dictionary `config`.
- `src/inference.py`: This file generates the prediction of individual components in the proposed model. Basically, it loads the model from the checkpoint and read data to produce the prediction, then save it.
- `experiment/04_ensemble.ipynb`: This file is for ensemble learning (SSE). To train an ensemble model, we first need to read prediction of base learners (generated by `src/inference.py`), and then build a classifier on top. There are some visualizations in this file as well.

The raw, preprocessed dataset along with the source code of this work are available at https://github.com/hoangntc/BERT_ABSA. There is a `README.md` file in the repository with instructions for reproducing results. Documentation is also included in main functions and classes.

3.2 Experiment Settings

Parameters Setting We use Stanford CoreNLP [19] to generate the dependency parse trees for each sentence. For word tokenization and word feature initialization, we employ uncased BERT-base from Google which contains 12 hidden layers and 768 hidden units for each layer. We keep the max sequence length at 100 since the length of the review is usually short. We randomly split the train data into 80/20 for training/validation and tune model hyperparameters by using the Ray Tune package [25] with the early stopping criteria based on the AUC metric of the validation set. After tuning, we choose batch size at 128, learning rate at 0.0001, search a dropout value in the range [0.01, 0.3], and the dimension of final embedding is 256. The loss is optimized by Adam optimizer [26] with default settings. We observe that the best performance is achieved after a few number of iterations.

Dataset. We use SemEval 2014 Task 4 [2] dataset to evaluate our proposed approach. This dataset includes a number of reviews in Laptop and Restaurant. In each such a review, there is a list of aspect terms and corresponding labeled polarities, which could be *positive*, *negative* or *neutral*. Table 1 shows the statistics of the dataset.

Datasets	Positive		Negative		Neutral	
	Train	Test	Train	Test	Train	Test
Restaurant	2164	728	807	196	637	196
Laptop	994	341	870	128	464	169

Table 1: Statistical information of datasets used in our experiments.

Baselines. We compare our proposed model (`SSE`) to the following baselines including (1) *Attention-based model*: `IAN` [27] and `PBAN` [15], (2) *Convolutional Neural Network-based model*: `CAPSNet` [28], (3) *Graph-based model*: `AS-CNN` [17], `AS-GCN` [17], and (4) *BERT-based and Graph-based Fusion model*: `RGAT` [29]. The results of baselines are from the published papers as all models are tested in the same set. We also conduct the experiments to check the performance of single components in our methods including `SEM`, `SYN` and `SS` as described in Section 2.3. It is worth to note that the first difference between our graph-based method and other graph-based methods are the initial word/subword embeddings.

Evaluation metrics. We use accuracy metric computed in the test set for evaluation.

Model	Restaurant	Laptop
IAN	79.30%	72.10%
PBAN	81.16%	74.12%
CAPSNet	78.80%	72.70%
AS-CNN	81.70%	72.60%
AS-GCN	80.80%	75.60%
R-GAT	83.30%	77.42%
R-GAT+BERT	86.60%	78.21%
SEM	83.91%	77.72%
SYN	84.66%	77.89%
SS	85.04%	78.99%
SSE	**86.70%**	**79.78%**

Table 2: Overall performance based on Accuracy.

3.3 Findings

Table 2 presents a comparison in terms of the accuracy between the baselines (first group) and our proposed model including its individual submodules (second group). We have following findings: (1) Our proposed model and its components achieve better performance compared to the baseline models except `RGAT` model. Especially, a standard BERT-based method can significantly increase the performance when compared with some graph-based models such as `AS-GCN` and `AS-GNN`. This suggests that the semantics information extracted by BERT plays a critical role in this task and it maybe relatively more important than syntactical information. (2) Comparing between `SEM` and `SYN`, we may conclude that the model is improved by a small margin when syntax information is involved in the learning process. (3) Combining the two types of feature set extracted by two components further improves the final results 2% for both datasets. Similar result is shown in `RGAT` model. (4) Among all graph-based methods, `RGAT` achieves the most competitive result. Besides, when comparing `RGAT+BERT` to our `SS` model, we may see that our model get a lower accuracy score in Restaurant dataset, indicating that the additional information of the dependency tree such as the dependency relation may help to enrich the graph embedding. (5) `SSE` model achieves the best performance among all methods. In addition, during training, we can observe that the ensemble learning also gives a more stable improvement compared to other methods, which suggests that adopting it may contribute significantly to sentiment analysis task.

4 Discussion

In this work, we propose an ensemble learning to combine information from semantic and syntax aware methods to enhance the performance of the neural network in the task of sentiment analysis. The improvement in our proposed model on the SemEval dataset show that semantic and syntactic information is of tremendous importance.

There are many things that we can do to extend our work. In our future work, we would like to explore further on how to extract the relationship between the aspect term and other terms especially in the case that the aspect term is mentioned multiple times in one review with different words. The future work can be: (1) applying co-reference resolution to identify the aspect term in the review, (2) modeling the relationship between the words mentioning the term and the remaining part of the sentence, and (3) using self-supervised learning to train the model to take advantage of available information such as part-of-speech or dependency relation.

References

[1] Bing Liu. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.

[2] Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. Attention based LSTM for target dependent sentiment classification. In Satinder P. Singh and Shaul Markovitch, editors,

[2]https://alt.qcri.org/semeval2014/task4/

Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA, pages 5013–5014. AAAI Press, 2017.

[3] Jiangming Liu and Yue Zhang. Attention modeling for targeted sentiment. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 572–577. Association for Computational Linguistics, 2017.

[4] Jiangfeng Zeng, Xiao Ma, and Ke Zhou. Enhancing attention-based LSTM with position context for aspect-level sentiment classification. *IEEE Access*, 7:20462–20471, 2019.

[5] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

[6] Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In Preslav Nakov and Torsten Zesch, editors, *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*, pages 437–442. The Association for Computer Linguistics, 2014.

[7] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27, 2011.

[8] Kang Liu, Heng Li Xu, Yang Liu, and Jun Zhao. Opinion target extraction using partially-supervised word alignment model. In Francesca Rossi, editor, *IJCAI*, pages 2134–2140. IJCAI/AAAI, 2013.

[9] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent Twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[10] Duy-Tin Vo and Yue Zhang. Target-dependent twitter sentiment classification with rich automatic features. In Qiang Yang and Michael J. Wooldridge, editors, *IJCAI*, pages 1347–1353. AAAI Press, 2015.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

[12] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[13] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. Effective lstms for target-dependent sentiment classification. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 3298–3307. ACL, 2016.

[14] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2048–2057. JMLR.org, 2015.

[15] Shuqin Gu, Lipeng Zhang, Yuexian Hou, and Yin Song. A position-aware bidirectional attention network for aspect-level sentiment analysis. In Emily M. Bender, Leon Derczynski, and Pierre Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 774–784. Association for Computational Linguistics, 2018.

[16] Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. Target-dependent sentiment classification with BERT. *IEEE Access*, 7:154290–154299, 2019.

[17] Chen Zhang, Qiuchi Li, and Dawei Song. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4567–4577. Association for Computational Linguistics, 2019.

[18] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 380–385. Association for Computational Linguistics, 2019.

[19] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[20] David Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 12 1992.

[21] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[23] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

[24] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[25] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[27] Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Interactive attention networks for aspect-level sentiment classification. *CoRR*, abs/1709.00893, 2017.

[28] Zhuang Chen and Tieyun Qian. Transfer capsule network for aspect level sentiment classification. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 547–556. Association for Computational Linguistics, 2019.

[29] Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. Relational graph attention network for aspect-based sentiment analysis. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3229–3238. Association for Computational Linguistics, 2020.