PyABSA: A Modularized Framework for Reproducible Aspect-based Sentiment Analysis

Heng Yang
hy345@exeter.ac.uk
Department of Computer,
University of Exeter
Exeter, UK

Chen Zhang
czhang@bit.edu.cn
School of Computer Science,
Beijing Institute of Technology
Beijing, China

Ke Li
k.li@exeter.ac.uk
Department of Computer,
University of Exeter
Exeter, UK

ABSTRACT

The advancement of aspect-based sentiment analysis (ABSA) has highlighted the lack of a user-friendly framework that can significantly reduce the difficulty of reproducing state-of-the-art ABSA performance, especially for beginners. To meet this demand, we present PyABSA, a modularized framework built on PyTorch for reproducible ABSA. To facilitate ABSA research, PyABSA supports several ABSA subtasks, including aspect term extraction, aspect sentiment classification, and end-to-end aspect-based sentiment analysis. With just a few lines of code, the result of a model on a specific dataset can be reproduced. With a modularized design, PyABSA can also be flexibly extended to incorporate new models, datasets, and other related tasks. Additionally, PyABSA highlights its data augmentation and annotation features, which significantly address data scarcity. The project is available at: https://github.com/yangheng95/PyABSA.

CCS CONCEPTS

• **Computing methodologies → Information extraction**.

KEYWORDS

Aspect-based sentiment analysis, data annotation tool, pretrained language model

ACM Reference Format:
Heng Yang, Chen Zhang, and Ke Li. 2023. PyABSA: A Modularized Framework for Reproducible Aspect-based Sentiment Analysis. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3583780.3614752

1 INTRODUCTION

Aspect-based sentiment analysis (ABSA) [15–17] has made remarkable strides in recent years, specifically in the subtasks of aspect term extraction (ATE) [7–9, 21–24, 28, 29], aspect sentiment classification (ASC) [2, 4, 6, 10–12, 14, 19, 20, 30, 31, 34], and end-to-end aspect-based sentiment analysis (E2EABSA) [27, 32, 33]. Take for example the sentence "I love the *pizza* at this restaurant, but the *service* is terrible." Here, there are two aspects - "*pizza*" and "*service*" - toward which the sentiments are positive and negative, respectively. ATE aims to extract the two aspects, ASC aims to detect the corresponding sentiments given the aspects, and E2EABSA[1] aims to perform the extraction and detection in one step.

Although an enormous number of models have been proposed in ABSA, they typically feature distinct architectures (e.g., LSTM, GCN, BERT) and optimizations (e.g., data pre-processing, evaluation metric). This complexity makes it challenging to reproduce their reported results, even when their code is available. To address this issue and promote fair comparison, we introduce PyABSA, a modularized framework built on PyTorch for reproducible ABSA. We provide a demonstration video[2] to showcase the basic usages of PyABSA.

PyABSA provides easy-to-use model training, evaluation, and inference on the aforementioned ABSA subtasks, supporting 31+ models and 30+ datasets. PyABSA allows beginners to reproduce the results of a model on a specific dataset with just a few lines of code[3]. In addition to using PyABSA to reproduce results, we have also released a series of trained checkpoints, which can be accessed through the Transformers Model Hub[4], offering exact reproducibility for users who need it.

Furthermore, PyABSA is a framework with a modularized organization. Technically, PyABSA comprises five major modules: the template class, configuration manager, dataset manager, metric visualizer, and checkpoint manager. This makes it flexible to extend the provided templates to considered models, datasets, and other related tasks with minor modifications.

It is widely recognized that ABSA models often suffer from a shortage of data and the absence of datasets in specific domains. Utilizing an ABSA-oriented data augmentor, we can provide up to $200K+$ additional examples per dataset. The augmented datasets can improve the accuracy of models by $1-3\%$. To encourage the community to contribute custom datasets, we provide a data annotation interface.

It's important to note that there are other existing projects partly achieving similar goals to PyABSA. However, the advantages of PyABSA over these projects can be distinguished in the following aspects:

- PyABSA democratizes reproducible ABSA research by supporting a larger array of models and datasets across primarily concerned ABSA subtasks.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614752

[1]There are aliases for ASC and E2EABSA in some research, i.e., APC and ATEPC.
[2]The video can be accessed at: https://www.youtube.com/watch?v=Od7t6CuCo6M
[3]We provide an inference service demo at https://huggingface.co/spaces/yangheng/Multilingual-Aspect-Based-Sentiment-Analysis
[4]The Model Hub of PyABSA is powered by Huggingface Space.

- PyABSA is a modularized framework that is flexible and can be extended to include considered models, datasets, and other related tasks due to its organization.
- PyABSA additionally offers data augmentation and data annotation features to address the data scarcity in ABSA.

2 SUPPORTED TASKS

We primarily support three subtasks in ABSA, namely ATE, ASC, E2EABSA, aspect sentiment triplet extraction(ATSC), and aspect category opinion sentiment quadruple extraction (ACOS). Each subtask contains its own models and datasets, which adds up to 31+ models and 30+ datasets in total.

Table 1: The prevalent models provided by PyABSA. ATE and E2EABSA share similar models. Note that the models based on BERT can be adapted to other pre-trained language models from HuggingFace Transformers.

Model	Task	Reference	GloVe	BERT
Fast-LSA-T	ASC / ATSC	Yang et al. [26]	✓	✓
Fast-LSA-S		Yang et al. [26]	✓	✓
Fast-LSA-P		Yang et al. [26]	✓	✓
BERT-ATESC	ATE / E2E	Devlin et al. [3]	✗	✓
Fast-LCF-ATESC		Yang et al. [27]	✗	✓
Fast-LCFS-ATESC		Yang et al. [27]	✗	✓
EMC-GCN	ATSC	Chen et al. [1]	✗	✓
ABSA-Instruction	ACOS	Scaria et al. [18]	✗	✓

2.1 Models & Datasets

The primary challenge in unifying different models into a single framework is accommodating the distinct architectures and optimizations used. We strive to bridge this gap with PyABSA, which has a large model pool covering attention-based, graph-based, and BERT-based models. The supported models are listed in Table 1.

PyABSA also consolidates a broad variety of datasets across various domains and languages, including laptops, restaurants, MOOCs, Twitter, and more. As far as we know, PyABSA maintains the largest collection of ABSA datasets, which can be viewed in Table 2. With just a few lines of code, researchers and users can utilize these built-in models and datasets for their own purposes. An example training pipeline for ASC is given in Snippet 1.

Snippet 1: The code snippet of an ASC training pipeline.

```
from pyabsa import AspectSentimentClassification as ASC

config = ASC.ASCConfigManager.get_asc_config_multilingual()
config.model = ASC.ASCModelList.FAST_LSA_T_V2

datasets_path = ASC.ABSADatasetList.Multilingual
sent_classifier = Trainer(config=config,
                          dataset=datasets_path,
                          ).load_trained_model()
```

2.2 Reproduction

We also present a preliminary performance overview of the models across the datasets provided in PyABSA. The results can be found in the GitHub repository. The standard deviations of the results are also included in parentheses. We used the collection of all datasets from PyABSA as the multilingual dataset. Please note that a "-" in the results table indicates that the graph-based models are not applicable to those specific datasets. The checkpoints of these models are also provided for exact reproducibility. An end-to-end (E2E) ABSA example inference pipeline is provided in Snippet2.

Table 2: A brief list of featured datasets in various languages provided by PyABSA.

Dataset	Language	# of Examples			Source
		Training Set	Validation Set	Testing Set	
Laptop14	English	2328	0	638	SemEval 2014
Restaurant14	English	3604	0	1120	SemEval 2014
MAMS	English	11181	1332	1336	Jiang et al. [5]
Yelp	English	808	0	245	WeiLi9811@GitHub
Phone	Chinese	1740	0	647	Peng et al. [13]
Car	Chinese	862	0	284	Peng et al. [13]
Notebook	Chinese	464	0	154	Peng et al. [13]
Camera	Chinese	1500	0	571	Peng et al. [13]
MOOC	Chinese	1583	0	396	jmc-123@GitHub
Arabic	Arabic	9620	0	2372	SemEval 2016
Dutch	Dutch	1283	0	394	SemEval 2016
Spanish	Spanish	1928	0	731	SemEval 2016
Turkish	Turkish	1385	0	146	SemEval 2016
Russian	Russian	3157	0	969	SemEval 2016
French	French	1769	0	718	SemEval 2016

Snippet 2: The code snippet of an E2EABSA inference pipeline.

```
from pyabsa import AspectTermExtraction as ATE

aspect_extractor = ATE.AspectExtractor(checkpoint="multilingual")

examples = ["But_the_staff_was_so_nice_to_us_."]
results = aspect_extractor.predict(example=examples) # simple inference
```

3 MODULARIZED FRAMEWORK

The main design of PyABSA is shown in Figure 1, which includes five necessary modules. We start by exploring task instances, which are abstracted as template classes. Afterward, we dive into other modules (i.e., configuration manager, dataset manager, metric visualizer, checkpoint manager), elaborating their roles in getting PyABSA modularized.

3.1 Template Classes

PyABSA streamlines the process of developing models for ABSA subtasks, with a range of templates (refer to the five template classes in Figure 1) that simplify the implementation of models and ease the customization of data.

We follow a software engineering design with common templates and interfaces, allowing users to define models with model utilities, process data with data utilities, train models with trainers, and infer models with predictors. These can be all achieved simply by inheriting the templates without manipulating the common modules. The inherited modules come with a uniform interface for all task-agnostic features.

3.2 Configuration Manager

Configuration manager handles environment configurations, model configurations, and hyperparameter settings. It extends the Python

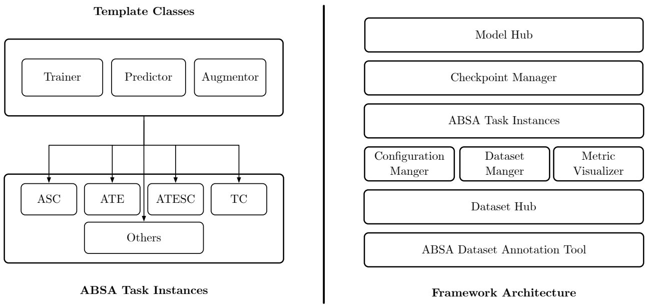


Figure 1: The left half of the diagram introduces the template classes provided in PyABSA. Typically, each ABSA subtask has five template classes that need to be instantiated, except for the augmenter, which is optional. The right side of the diagram displays the main framework of PyABSA. At the lowest level is the data annotation, which is suitable for creating custom datasets, and the created datasets can be shared on the dataset hub. The three modules in the middle are the generic modules, which are designed for training based on new datasets or models. The checkpoint manager is used to connect to the model hub. It is responsible for uploading and downloading models, as well as instantiating inference models.

Namespace object for improving user-friendliness. Additionally, The configuration manager possesses a configuration checker to make sure that incorrect configurations do not pass necessary sanity checks, helping users keep track of their training settings.

3.3 Dataset Manager

Dataset manager enables users to manage a wide range of built-in and custom datasets. Each dataset is assigned with unique ID and name for management, and the dataset items are designed as nest objects to enhance flexibility. This design makes it simple to combine datasets for ensemble learning and multilingual ABSA tasks. Moreover, the dataset manager also takes care of seamlessly connecting to the ABSA dataset hub, automatically downloading and managing the integrated datasets.

3.4 Metric Visualizer

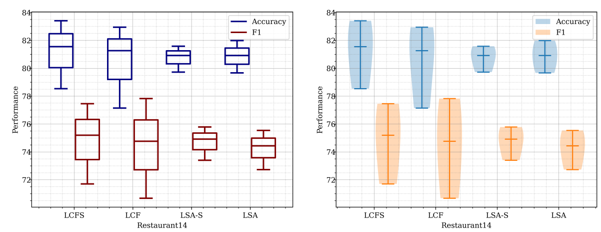
As a vital effort towards streamlined evaluation and fair comparisons, metric visualizer[5] for PyABSA to automatically record, manage, and visualize various metrics (such as Accuracy, F-measure, STD, IQR, etc.). The metric visualizer can track metrics in real-time or load saved metrics records and produce box plots, violin plots, trajectory plots, Scott-Knott test plots, significance test results, etc. An example of auto-generated visualizations is shown in Figure 2 and more plots and experiment settings can be found in the GitHub repository. The metric visualizer streamlines the process of visualizing performance metrics and eliminates potential biases in metric statistics.

[5]The metric visualizer was developed specifically for PyABSA and is available as an independent open-source project at: https://github.com/yangheng95/metric-visualizer

Figure 2: The metrics summary and a part of automatic visualizations processed by metric visualizer in PyABSA. The experimental dataset is ARTS-Laptop14, an adversarial dataset for ASC.

------------------------------------- Metric Visualizer -------------------------------------

Metric	Trial	Values	Average	Median	Std	IQR	Min	Max
Accuracy	LCFS	[83.86, 84.01, 83.39]	83.75	83.86	0.27	0.31	83.39	84.01
Accuracy	LCF	[83.23, 82.76, 83.23]	83.07	83.23	0.22	0.24	82.76	83.23
Accuracy	LSA-S	[82.29, 82.6, 83.7]	82.86	82.6	0.6	0.71	82.29	83.7
Accuracy	LSA-T	[82.76, 82.76, 83.7]	83.07	82.76	0.44	0.47	82.76	83.7
macro F1	LCFS	[80.9, 80.87, 80.4]	80.72	80.87	0.23	0.25	80.4	80.9
macro F1	LCF	[80.36, 80.08, 80.59]	80.34	80.36	0.21	0.25	80.08	80.59
macro F1	LSA-S	[78.89, 79.48, 80.3]	79.56	79.48	0.58	0.71	78.89	80.3
macro F1	LSA-T	[79.79, 79.25, 80.53]	79.85	79.79	0.53	0.64	79.25	80.53



3.5 Checkpoint Manager

The checkpoint manager manages the trained model checkpoints and interacts with the model hub. Users can easily query available checkpoints for different ABSA subtasks and instantiate an inference model by specifying its checkpoint name. Users can query available checkpoints in a few lines of code from the model hub. The example of available checkpoints is shown in Figure 3. Although

Figure 3: A part of available checkpoints for E2E ABSA in PyABSA's model hub.

```
********** Available E2E ABSA model checkpoints for Version:2.0.29a0 (this version) **********
------------------------------------------------------------------------------------------------
Checkpoint Name: multilingual
Training Model: FAST-LCF-ATEPC
Training Dataset: ABSADatasets.Multilingual
Language: Multilingual
Description: Trained on RTX3090
Available Version: 1.16.0+
Checkpoint File: fast_lcf_atepc_Multilingual_cdw_apcacc_80.81_apcf1_73.75_atef1_76.01.zip
Author: H, Yang (hy345@exeter.ac.uk)
```

connecting to the model hub is the most convenient way to get an inference model, we also provide two alternative ways:

- Searching for trained or cached checkpoints using keywords or paths through the checkpoint manager.
- Building inference models using trained models returned by the trainers, which eliminates the need for saving checkpoints to disk.

The checkpoint manager for any subtask is compatible with GloVe and pre-trained models based on transformers, and with the help of PyABSA's interface, launching an ATESC service requires just a few lines of code.

4 FEATURED FUNCTIONALITIES

4.1 Data Augmentation

In ABSA, data scarcity can lead to inconsistencies in performance evaluation and difficulties with generalizing across domains. To address this issue, PyABSA has adopted an automatic text augmentation method, i.e., BoostAug [25]. This method balances diversity and skewness in the distribution of augmented data. In our experiments, the text augmentation method significantly boosted the classification accuracy and F1 scores of all datasets and models, whereas previous text augmentation techniques had a negative impact on model performance.

4.2 Dataset Annotation

Annotating ABSA datasets is more difficult compared to pure text classification. As there is no open-source tool available for annotating ABSA datasets, creating custom datasets becomes a critical challenge. In PyABSA, we have got users rescued by providing a manual annotation interface[6] contributed by the community, along with an automatic annotation interface.

Manual Annotation. To ensure accurate manual annotation, our contributor developed a specialized ASC annotation tool for PyABSA. This tool runs on web browsers, making it easy for anyone to create their own datasets with just a web browser. The annotation tool outputs datasets for various ABSA sub-tasks, such as ASC and ATESC sub-tasks, and we even provide an interface to help users convert datasets between different sub-tasks.

Automatic Annotation. To make manual annotation easier and address the issue of limited data, we offer an automatic annotation method in PyABSA. This interface is powered by a trained E2EABSA model and uses a hub-powered inference model to extract aspect terms and sentiment polarities. It enables users to quickly expand small datasets with annotated ABSA instances. Check out the following example for a demonstration of the automatic annotation interface:

Snippet 3: The code snippet of automatic annotation.

```
from pyabsa import make_ABSA_dataset

# annotate "raw_data" using the "multilingual" ATESC model
make_ABSA_dataset(
    dataset_name_or_path='raw_data',
    checkpoint='multilingual'
)
```

Ensemble Training. The model ensemble is a crucial technique in deep learning, and it is common to enhance ABSA performance in real-world projects through the model ensemble. To simplify the process for users, PyABSA provides an easy-to-use model ensemble feature without any code changes. Furthermore, PyABSA offers convenient ensemble methods for users to effortlessly augment their training data using built-in datasets from the data center. For example, when PyABSA recognizes a model or dataset as a list, it will automatically perform an ensemble. We showcase this simple ensemble method in Snippet 4.

[6] https://github.com/yangheng95/ABSADatasets/tree/v2.0/DPT

Snippet 4: The training code snippet of models ensemble in PyABSA.

```
from pyabsa import AspectPolarityClassification as APC

models = [
        ASC.ASCModelList.FAST_LSA_T_V2,
        ASC.ASCModelList.FAST_LSA_S_V2,
        ASC.ASCModelList.BERT_SPC_V2,
    ]
config = ASC.ASCConfigManager.get_apc_config_english()
config.model = models
```

Ensemble Inference. PyABSA includes an ensemble inference module for all subtasks, which enables users to aggregate the results of multiple models to produce a final prediction, thereby leveraging the strengths of each individual model and resulting in improved performance and robustness compared to using a single model alone. We provide an example of ensemble inference in Snippet 5.

Snippet 5: The inference code snippet of models ensemble in PyABSA.

```
from pyabsa.utils import VoteEnsemblePredictor

checkpoints = {
    ckpt: APC.SentimentClassifier(checkpoint=ckpt)
    for ckpt in findfile.find_cwd_dirs(or_key=["laptop14"])
}
ensemble_predictor = VoteEnsemblePredictor(
    checkpoints,
    weights=None,
    numeric_agg="mean",
    str_agg="max_vote"
)
ensemble_predictor.predict("The␣[B-ASP]food[E-ASP]␣was␣good!")
```

5 CONCLUSIONS AND FUTURE WORK

We present PyABSA, a modularized framework for reproducible ABSA. Our goal was to democratize the reproduction of ABSA models with a few lines of code and provide an opportunity of implementing ideas with minimal modifications on our prototypes. Additionally, the framework comes equipped with powerful data augmentation and annotation features, largely addressing the data insufficiency of ABSA. In the future, we plan to expand the framework to include other ABSA subtasks, such as aspect sentiment triplet extraction.

ACKNOWLEDGEMENTS

This work was supported in part by the UKRI Future Leaders Fellowship under Grant MR/S017062/1 and Grant MR/X011135/1; in part by NSFC under Grant 62076056; in part by the Royal Society under Grant IES/R2/212077; in part by EPSRC under Grant 2404317; and in part by the Amazon Research Award. We appreciate all contributors who help PyABSA e.g., committing code or datasets; the community's support makes PyABSA even better. Furthermore, we appreciate all ABSA researchers for their open-source models that improve ABSA.

REFERENCES

[1] Hao Chen, Zepeng Zhai, Fangxiang Feng, Ruifan Li, and Xiaojie Wang. 2022. Enhanced Multi-Channel Graph Convolutional Network for Aspect Sentiment Triplet Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*. Association for Computational Linguistics, 2974–2985. https://doi.org/10.18653/v1/2022.acl-long.212

[2] Junqi Dai, Hang Yan, Tianxiang Sun, Pengfei Liu, and Xipeng Qiu. 2021. Does syntax matter? A strong baseline for Aspect-based Sentiment Analysis with RoBERTa. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Association for Computational Linguistics, 1816–1829. https://doi.org/10.18653/v1/2021.naacl-main.146

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *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)*. Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[4] Binxuan Huang and Kathleen M. Carley. 2019. Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks. In *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*. Association for Computational Linguistics, 5468–5476. https://doi.org/10.18653/v1/D19-1549

[5] Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. 2019. A Challenge Dataset and Effective Models for Aspect-Based Sentiment Analysis. In *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*. Association for Computational Linguistics, 6279–6284. https://doi.org/10.18653/v1/D19-1654

[6] Ruifan Li, Hao Chen, Fangxiang Feng, Zhanyu Ma, Xiaojie Wang, and Eduard H. Hovy. 2021. Dual Graph Convolutional Networks for Aspect-based Sentiment Analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*. Association for Computational Linguistics, 6319–6329. https://doi.org/10.18653/v1/2021.acl-long.494

[7] Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018. Aspect Term Extraction with History Attention and Selective Transformation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. ijcai.org, 4194–4200. https://doi.org/10.24963/ijcai.2018/583

[8] Xin Li and Wai Lam. 2017. Deep Multi-Task Learning for Aspect Term Extraction with Memory Interaction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics, 2886–2892. https://doi.org/10.18653/v1/d17-1310

[9] Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. Exploring Sequence-to-Sequence Learning in Aspect Term Extraction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 3538–3547. https://doi.org/10.18653/v1/p19-1344

[10] Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive Attention Networks for Aspect-Level Sentiment Classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. ijcai.org, 4068–4074. https://doi.org/10.24963/ijcai.2017/568

[11] Fang Ma, Chen Zhang, and Dawei Song. 2021. Exploiting Position Bias for Robust Aspect Sentiment Classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 1352–1358.

[12] Fang Ma, Chen Zhang, Bo Zhang, and Dawei Song. 2022. Aspect-specific context modeling for aspect-based sentiment analysis. In *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 513–526.

[13] Haiyun Peng, Yukun Ma, Yang Li, and Erik Cambria. 2018. Learning multi-grained aspect target sequence for Chinese sentiment analysis. *Knowl. Based Syst.* 148 (2018), 167–176. https://doi.org/10.1016/j.knosys.2018.02.034

[14] Minh Hieu Phan and Philip O. Ogunbona. 2020. Modelling Context and Syntactical Features for Aspect-based Sentiment Analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 3211–3220. https://doi.org/10.18653/v1/2020.acl-main.293

[15] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia V. Loukachevitch, Evgeniy V. Kotelnikov, Núria Bel, Salud María Jiménez Zafra, and Gülsen Eryigit. 2016. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. The Association for Computer Linguistics, 19–30. https://doi.org/10.18653/v1/s16-1002

[16] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 Task 12: Aspect Based Sentiment Analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*. The Association for Computer Linguistics, 486–495. https://doi.org/10.18653/v1/s15-2082

[17] 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@COLING 2014, Dublin, Ireland, August 23-24, 2014*. The Association for Computer Linguistics, 27–35. https://doi.org/10.3115/v1/s14-2004

[18] Kevin Scaria, Himanshu Gupta, Saurabh Arjun Sawant, Swaroop Mishra, and Chitta Baral. 2023. InstructABSA: Instruction Learning for Aspect Based Sentiment Analysis. *CoRR* abs/2302.08624 (2023). https://doi.org/10.48550/arXiv.2302.08624 arXiv:2302.08624

[19] Yuanhe Tian, Guimin Chen, and Yan Song. 2021. Aspect-based Sentiment Analysis with Type-aware Graph Convolutional Networks and Layer Ensemble. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Association for Computational Linguistics, 2910–2922. https://doi.org/10.18653/v1/2021.naacl-main.231

[20] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, and Yi Chang. 2021. Eliminating Sentiment Bias for Aspect-Level Sentiment Classification with Unsupervised Opinion Extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*. Association for Computational Linguistics, 3002–3012. https://doi.org/10.18653/v1/2021.findings-emnlp.258

[21] Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive Neural Conditional Random Fields for Aspect-based Sentiment Analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics, 616–626. https://doi.org/10.18653/v1/d16-1059

[22] Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled Multi-Layer Attentions for Co-Extraction of Aspect and Opinion Terms. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. AAAI Press, 3316–3322. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14441

[23] Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double Embeddings and CNN-based Sequence Labeling for Aspect Extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*. Association for Computational Linguistics, 592–598. https://doi.org/10.18653/v1/P18-2094

[24] Heng Yang. 2019. *PyABSA: Open Framework for Aspect-based Sentiment Analysis*. https://github.com/yangheng95/PyABSA

[25] Heng Yang and Ke Li. 2023. Boosting Text Augmentation via Hybrid Instance Filtering Framework. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 1652–1669. https://doi.org/10.18653/v1/2023.findings-acl.105

[26] Heng Yang, Biqing Zeng, Mayi Xu, and Tianxing Wang. 2021. Back to Reality: Leveraging Pattern-driven Modeling to Enable Affordable Sentiment Dependency Learning. *CoRR* abs/2110.08604 (2021). arXiv:2110.08604 https://arxiv.org/abs/2110.08604

[27] Heng Yang, Biqing Zeng, Jianhao Yang, Youwei Song, and Ruyang Xu. 2021. A multi-task learning model for Chinese-oriented aspect polarity classification and aspect term extraction. *Neurocomputing* 419 (2021), 344–356. https://doi.org/10.1016/j.neucom.2020.08.001

[28] Yunyi Yang, Kun Li, Xiaojun Quan, Weizhou Shen, and Qinliang Su. 2020. Constituency Lattice Encoding for Aspect Term Extraction. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. International Committee on Computational Linguistics, 844–855. https://doi.org/10.18653/v1/2020.coling-main.73

[29] Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised Word and Dependency Path Embeddings for Aspect Term Extraction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. IJCAI/AAAI Press, 2979–2985. http://www.ijcai.org/Abstract/16/423

[30] Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks. In *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*. Association for Computational Linguistics, 4567–4577. https://doi.org/10.18653/v1/D19-1464

[31] Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Syntax-aware aspect-level sentiment classification with proximity-weighted convolution network. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 1145–1148.

[32] Chen Zhang, Qiuchi Li, Dawei Song, and Benyou Wang. 2020. A Multi-task Learning Framework for Opinion Triplet Extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 819–828.

[33] Chen Zhang, Lei Ren, Fang Ma, Jingang Wang, Wei Wu, and Dawei Song. 2022. Structural Bias for Aspect Sentiment Triplet Extraction. In *Proceedings of the 29th International Conference on Computational Linguistics*. 6736–6745.

[34] Pinlong Zhao, Linlin Hou, and Ou Wu. 2020. Modeling sentiment dependencies with graph convolutional networks for aspect-level sentiment classification. *Knowl. Based Syst.* 193 (2020), 105443. https://doi.org/10.1016/j.knosys.2019.105443