

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

Heng Yang¹, Chen Zhang², Ke Li¹

¹Department of Computer Science, University of Exeter, EX4 4QF, Exeter, UK

²School of Computer Science, Beijing Institute of Technology, Beijing, China
{hy345, k.li}@exeter.ac.uk, czhang@bit.edu.cn

Abstract

The advancement of aspect-based sentiment analysis (ABSA) has urged the lack of a user-friendly framework that can largely lower the difficulty of reproducing state-of-the-art ABSA performance, especially for beginners. To meet the 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. Concretely, PyABSA integrates 29 models and 26 datasets. 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 considered models, datasets, and other related tasks. Besides, PyABSA highlights its data augmentation and annotation features, which significantly address data scarcity. All are welcome to have a try at <https://github.com/yangheng95/PyABSA>.

1 Introduction

Aspect-based sentiment analysis (ABSA) (Pontiki et al., 2014, 2015, 2016) has made remarkable strides in recent years, particularly in the subtasks of aspect term extraction (ATE) (Yin et al., 2016; Wang et al., 2016a; Li and Lam, 2017; Wang et al., 2017; Li et al., 2018b; Xu et al., 2018; Ma et al., 2019; Yang, 2019; Yang et al., 2020), aspect sentiment classification (ASC) (Ma et al., 2017; Zhang et al., 2019; Huang and Carley, 2019; Phan and Ogunbona, 2020; Zhao et al., 2020; Li et al., 2021a; Dai et al., 2021; Tian et al., 2021; Wang et al., 2021), and end-to-end aspect-based sentiment analysis (E2EABSA) (Yang et al., 2021b). In an example sentence that “I love the *pizza* at this restaurant, but the *service* is terrible.”, there are two aspects “*pizza*” and “*service*”. towards which the sentiments are positive and negative, respectively. Here,

ATE aims to extract the two aspects, ASC aims to detect the corresponding sentiments given the aspects, and E2EABSA¹ aims to achieve the extraction and detection as one.

While an enormous number of models have been proposed in ABSA, however, they typically have distinguished architectures (e.g., LSTM, GCN, BERT) and optimizations (e.g., data pre-processing, evaluation metric), making it hard to reproduce their reported results even if their code is released. To address this issue and promote a fair comparison, we introduce PyABSA, a modularized framework built on PyTorch for reproducible ABSA. We provide a demonstration video² to show the basic usages of PyABSA.

PyABSA enables easy-to-use model training, evaluation, and inference on aforementioned ABSA subtasks with 29 models and 26 datasets supported. PyABSA allows beginners to reproduce the result of a model on a specific dataset with just a few lines of code. In addition to using PyABSA to reproduce results, we have also released a range of trained checkpoints, which can be accessed through the Transformers Model Hub³ for users who need exact reproducibility.

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

It is widely recognized that ABSA models suffers from the shortage of data and the absence of datasets in specific domains. Utilizing an ABSA-oriented data augmentor, we are able to provide up

¹There are aliases for ASC and E2EABSA in some research, i.e., APC and ATEPC.

²The video can be accesses at: <https://www.youtube.com/watch?v=Od7t6CuCo6M>

³The [Model Hub](#) of PyABSA is powered by Huggingface Space.

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 an data annotation interface.

It is noteworthy that there are other existing projects partly achieving similar goals with PyABSA. We should mark that the advantages of PyABSA over these projects are in the following aspects.

- PyABSA democratizes reproducible ABSA research by supporting a larger array of models and datasets among mainly concerned ABSA subtasks.
- PyABSA is a modularized framework that is flexible to be extended to considered models, datasets, and other related tasks thanks to its organization.
- PyABSA additionally offers data augmentation and data annotation features to address the data scarcity in ABSA.

2 Supported Tasks

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
AOA	ASC	Huang et al. (2018)	✓	✓
ASGCN		Zhang et al. (2019)	✓	✓
ATAE-LSTM		Wang et al. (2016b)	✓	✓
Cabasc		Liu et al. (2018)	✓	✓
IAN		Ma et al. (2017)	✓	✓
LSTM-ASC		Hochreiter et al. (1997)	✓	✓
MemNet		Tang et al. (2016b)	✓	✓
MGAN		Fan et al. (2018)	✓	✓
RAM		Chen et al. (2017)	✓	✓
TC-LSTM		Tang et al. (2016a)	✓	✓
TD-LSTM		Tang et al. (2016a)	✓	✓
TNet-LF		Li et al. (2018a)	✓	✓
BERT-ASC		Devlin et al. (2019)	✗	✓
BERT-SPC		Devlin et al. (2019)	✗	✓
DLCF-DCA		Xu et al. (2022)	✗	✓
DLCFS-DCA		Xu et al. (2022)	✗	✓
Fast-LCF-ASC		Zeng et al. (2019)	✗	✓
Fast-LCFS-ASC		Zeng et al. (2019)	✗	✓
LCA-BERT		Yang and Zeng (2020)	✗	✓
LCF-BERT		Zeng et al. (2019)	✗	✓
LCFS-BERT		Zeng et al. (2019)	✗	✓
Fast-LSA-T		Yang et al. (2021a)	✗	✓
Fast-LSA-S		Yang et al. (2021a)	✗	✓
Fast-LSA-P		Yang et al. (2021a)	✗	✓
BERT-ATESC	ATE / E2E	Devlin et al. (2019)	✗	✓
Fast-LCF-ATESC		Yang et al. (2021b)	✗	✓
Fast-LCFS-ATESC		Yang et al. (2021b)	✗	✓
LCF-ATESC		Yang et al. (2021b)	✗	✓
LCFS-ATESC		Yang et al. (2021b)	✗	✓

We primarily support three subtasks in ABSA, namely ATE, ASC, and E2EABSA. Each subtask contains its own models and datasets, which adds

up to 29 models and 26 datasets in total.

2.1 Models & Datasets

The core difficulty in unifying different models into a framework is that distinguished architectures and optimizations being used. We strive to bridge the gap in PyABSA, which has to our best knowledge the largest model pool covering attention-based, graph-based, and BERT-based models, etc. The supported models are listed in Table 1.

PyABSA also gathers a wide variety of datasets across various domains and languages, including laptops, restaurants, MOOCs, Twitter, and others. As far as we know, PyABSA maintains the largest ever number of ABSA datasets, which can be viewed in Table 2.

With just a few lines of code, researchers and users can invoke these builtin models and datasets for their own purposes. An example training pipeline of 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,
                          checkpoint_save_mode=1, # save
                                                  state_dict instead of model
                          auto_device=True, # auto-select
                                                  cuda device
                          load_aug=True, # training using
                                                  augmentation data
                          ).load_trained_model()
```

2.2 Reproduction

We as well present a preliminary performance overview of the models over the datasets provided in PyABSA. The results, which are based on ten epochs of training using the configurations for reproduction, can be found in Appendix B. The standard deviations of the results are also attached in parentheses. We used the pile of all datasets from PyABSA as the multilingual one. Please note that "-" in the results table means that the graph-based models are not applicable for those specific datasets. The checkpoints of these models are also offered for exact reproducibility. An E2E ABSA example inference pipeline is given in Snippet 2.

Table 2: A list of datasets in various languages presented in PyABSA, where the datasets marked with [†] are used for adversarial research. The increased number of examples in the training set have been generated using our own ABSA automatic augmentation tool.

Dataset	Language	# of Examples			# of Augmented Examples	Source
		Training Set	Validation Set	Testing Set	Training Set	
Laptop14	English	2328	0	638	13325	SemEval 2014
Restaurant14	English	3604	0	1120	19832	SemEval 2014
Restaurant15	English	1200	0	539	7311	SemEval 2015
Restaurant16	English	1744	0	614	10372	SemEval 2016
Twitter	English	5880	0	654	35227	Dong et al. (2014)
MAMS	English	11181	1332	1336	62665	Jiang et al. (2019)
Television	English	3647	0	915	25676	Mukherjee et al. (2021)
T-shirt	English	1834	0	465	15086	Mukherjee et al. (2021)
Yelp	English	808	0	245	2547	WeiLi9811@GitHub
Phone	Chinese	1740	0	647	0	Peng et al. (2018)
Car	Chinese	862	0	284	0	Peng et al. (2018)
Notebook	Chinese	464	0	154	0	Peng et al. (2018)
Camera	Chinese	1500	0	571	0	Peng et al. (2018)
MOOC	Chinese	1583	0	396	0	jmc-123@GitHub
Shampoo	Chinese	6810	0	915	0	brightgems@GitHub
MOOC-En	English	1492	0	459	10562	aparnavalli@GitHub
Arabic	Arabic	9620	0	2372	0	SemEval 2016
Dutch	Dutch	1283	0	394	0	SemEval 2016
Spanish	Spanish	1928	0	731	0	SemEval 2016
Turkish	Turkish	1385	0	146	0	SemEval 2016
Russian	Russian	3157	0	969	0	SemEval 2016
French	French	1769	0	718	0	SemEval 2016
ARTS-Laptop14 [†]	English	2328	638	1877	13325	Xing et al. (2020)
ARTS-Restaurant14 [†]	English	3604	1120	3448	19832	Xing et al. (2020)
Kaggle [†]	English	3376	0	866	0	Khandeka@Kaggle
Chinese-Restaurant [†]	Chinese	26119	3638	7508	0	Zhang et al. (2022)

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

```

from pyabsa import AspectTermExtraction as ATE

aspect_extractor = ATE.AspectExtractor(
    "multilingual",
    data_num=100,
)
# simple inference
examples = [
    "But_the_staff_was_so_nice_to_us.",
    "But_the_staff_was_so_horrible_to_us.",
]
result = aspect_extractor.predict(
    example=examples,
    print_result=True, # print results in console
    ignore_error=True, # ignore an invalid input
    eval_batch_size=32, # set batch size
)
# batch inference
atepc_result = aspect_extractor.batch_predict(
    inference_source,
    save_result=False,
    print_result=True,
    pred_sentiment=True,
    eval_batch_size=32,
)

```

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. Afterwards, 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 defining models with model utilities, processing data with data utilities, training models with trainer, and inferring 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 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 in track of their training settings.

3.3 Dataset Manager

Dataset manager enables users to manage a wide range of builtin and custom datasets. Each dataset

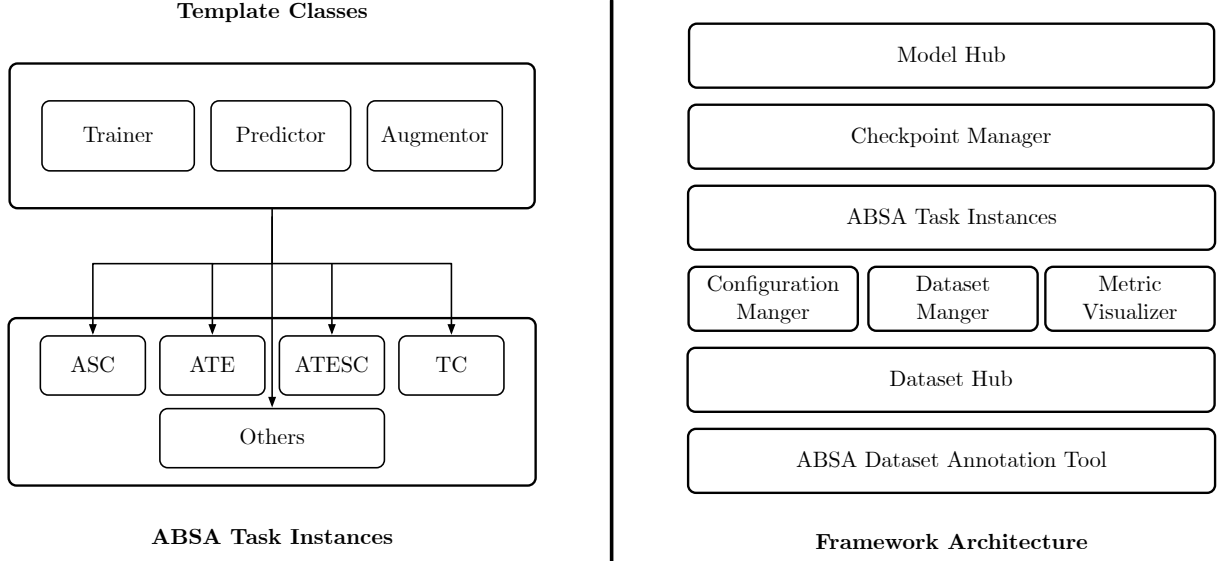


Figure 1: The left half of the diagram introduces the template classes provided in PyABSA. Typically, each ABSA subtask has 5 template classes that need to be instantiated, except for the augmenter which is optional. The right side of the diagram shows the main framework of PyABSA. The lowest level is the data annotation, which is suitable for creating custom datasets and the created datasets can be shared to the dataset hub. The three modules in the middle are the generic modules, which are suitable for training based on new datasets or models. The checkpoint manager is used to connect to the model hub and is responsible for uploading and downloading models and instantiating inference models.

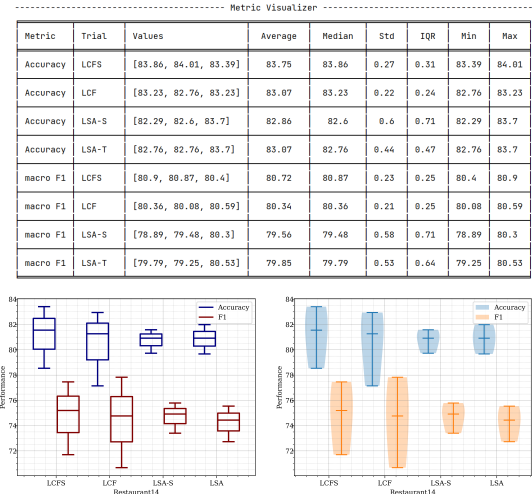
is assigned with unique ID and name for management, and the dataset items are designed as nest objects to enhance flexibility. This design makes it simple to combine datasets for ensemble learning and multilingual ABSA tasks. Moreover, the dataset manager also takes care of seamlessly connect to the 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 visualizer⁴ 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 Appendix C. The metric visualizer streamlines the process of visualizing performance metrics and eliminates potential biases in metric statistics.

⁴The 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.



3.5 Checkpoint Manager

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

Figure 3: The community-contributed manual dataset annotation tool provided for PyABSA.

Save your work

X

1
Positive
Neutral
Negative
Clear

ID	Text	Overall
1	cute animals. bought annual pass, and we have visited 3x now.	Select... v
2	fantastic place to go with the whole family. the tickets are slightly on the higher side but the experience is worth it. the animals appear to be well looked after and cared for. there is heaps of animals that roam freely that you can touch, pat and feed. im very impressed overall. the cafe is also very lovely with amazing food.	Select... v
3	we were up here mid february. weather was good for walking, and theres lots of it, so many paths and trails. was good to see so many open areas for the animals. kids enjoyed feeding the kangaroos and an attempt at an emu before nerves kicked in size difference walkways are well maintained and clean, but it is slightly hilly in some parts. spent about 2.5hrs walking around exploring everything. good for all ages.	Select... v
4	great place any weather. lots to see and do cafe, staff and gift shop are fantastic. become a member, visit twice in a year and it has paid for itself. ive been 4 times this year already	Select... v
5	a must visit for tourists and locals. we hadnt visited for many years and decided it was time to revisit. we bought yearly memberships so we can now visit whenever we like. well worth the money and value received with only 2 visits. the park is well laid out, paths are sealed so good for parks, wheelchairs, etc. all the animals were very relaxed and friendly. great views over adelaide by the yellow footed rock wallabies rocky. the cafe was inviting and served amazing wedges and good coffee. there is a cosy lo9king fireplace for the winter visits. the animals were quite active on a cooler summers morning.	Select... v

1

can query available checkpoints in few lines of code as in Snippet 3 from the model hub. The example of available checkpoints is shown in Figure 4.

Snippet 3: The code snippet of available checkpoints.

```
from pyabsa import available_checkpoints
from pyabsa import TaskCodeOption

checkpoint_map = available_checkpoints(
    # the code of ASC
    TaskCodeOption.Aspect_Polarity_Classification,
    show_ckpts=True
)
```

While 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 ATEPC 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 augmen-

Figure 4: 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_88.81_apcf1_73.75_atef1_76.81.zip
Author: H, Yang (hy345@exeter.ac.uk)
-----
Checkpoint Name: multilingual2
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_78.08_apcf1_77.81_atef1_75.41.zip
Author: H, Yang (hy345@exeter.ac.uk)
-----
Checkpoint Name: english
Training Model: FAST-LCF-ATEPC
Training Dataset: ATEPCDatasetList.English
Language: English
Description: Trained on RTX3090
Available Version: 1.10.5+
Checkpoint File: fast_lcf_atepc_English_cdw_apcacc_82.36_apcf1_81.89_atef1_75.43.zip
Author: H, Yang (hy345@exeter.ac.uk)
-----
```

tation method, i.e., BoostAug. 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. We refer a comprehensive overview of this text augmentation method to Yang and Li (2022).

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 criti-

cal challenge. In PyABSA, we have got users rescued by provide a manual annotation interface contributed by the community (referred to as Figure 3), along with an automatic annotation interface.

Manual Annotation To ensure accurate manual annotation, our contributor developed a specialized ASC annotation tool⁵ 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 ATESE sub-tasks, and we even provide an interface to help users convert datasets between different sub-tasks. Check out the community-contributed manual dataset annotation tool in Figure 3

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 4: The code snippet of automatic annotation.

```
from pyabsa import make_ABSA_dataset

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

Ensemble Training In deep learning, model ensemble is a crucial technique, and it is common to enhance ABSA performance in real-world projects through model ensemble. To simplify the process for users, PyABSA provides easy-to-use model ensemble 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 ensemble. We showcase this simple ensemble method in Snippet 5.

Ensemble Inference PyABSA includes an ensemble inference module for all subtasks, which enables users to aggregate the results of multiple

Snippet 5: The code snippet of an model ensemble in PyABSA.

```
import random
from pyabsa import (
    AspectSentimentClassification as ASC,
    ModelSaveOption,
    DeviceTypeOption
)

models = [
    ASC.ASCModelList.FAST_LSA_T_V2,
    ASC.ASCModelList.FAST_LSA_S_V2,
    ASC.ASCModelList.BERT_SPC_V2,
]

datasets = [
    ASC.ASCDatasetList.Laptop14,
    ASC.ASCDatasetList.Restaurant14,
    ASC.ASCDatasetList.Restaurant15,
    ASC.ASCDatasetList.Restaurant16,
    ASC.ASCDatasetList.MAMS
]

config = ASC.ASCConfigManager.get_apc_config_english()
config.model = models
config.pretrained_bert = 'roberta-base'
config.seed = [random.randint(0, 10000) for _ in range(3)]

trainer = ASC.ASCTrainer(
    config=config,
    dataset=datasets,
    checkpoint_save_mode=ModelSaveOption.
        SAVE_MODEL_STATE_DICT,
    auto_device=DeviceTypeOption.AUTO,
)
trainer.load_trained_model()
```

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 6.

Snippet 6: The code snippet of an model ensemble in PyABSA.

```
from pyabsa.utils import VoteEnsemblePredictor

checkpoints = {
    ckpt: APC.SentimentClassifier(checkpoint=ckpt)
    # use the findfile module to search all available
    checkpoints
    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

⁵<https://github.com/yangheng95/ABSADatasets/DPT>

and annotation features, largely addressing the data scarcity of ABSA. In the future, we plan to expand the framework to include other ABSA subtasks, such as aspect sentiment triplet extraction.

Acknowledgements

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

- Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. [Recurrent attention network on memory for aspect sentiment analysis](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 452–461. Association for Computational Linguistics.
- Scala Consultants. 2020. [Aspect-Based-Sentiment-Analysis](#).
- 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*, pages 1816–1829. Association for Computational Linguistics.
- 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)*, pages 4171–4186. Association for Computational Linguistics.
- Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. [Adaptive recursive neural network for target-dependent twitter sentiment classification](#). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 49–54. The Association for Computer Linguistics.
- Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. [Multi-grained attention network for aspect-level sentiment classification](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3433–3442. Association for Computational Linguistics.
- Sepp Hochreiter, Jürgen Schmidhuber, and Padding. 1997. [Long short-term memory](#). *Neural Comput.*, 9(8):1735–1780.
- 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*, pages 5468–5476. Association for Computational Linguistics.
- Binxuan Huang, Yanglan Ou, and Kathleen M. Carley. 2018. [Aspect level sentiment classification with attention-over-attention neural networks](#). In *Social, Cultural, and Behavioral Modeling - 11th International Conference, SBP-BRIMS 2018, Washington, DC, USA, July 10-13, 2018, Proceedings*, volume 10899 of *Lecture Notes in Computer Science*, pages 197–206. Springer.
- 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*, pages 6279–6284. Association for Computational Linguistics.
- Ruifan Li, Hao Chen, Fangxiang Feng, Zhanyu Ma, Xiaojie Wang, and Eduard H. Hovy. 2021a. [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*, pages 6319–6329. Association for Computational Linguistics.
- Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018a. [Transformation networks for target-oriented sentiment classification](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 946–956. Association for Computational Linguistics.
- Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018b. [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*, pages 4194–4200. ijcai.org.
- 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, pages 2886–2892. Association for Computational Linguistics.
- Zhengyan Li, Yicheng Zou, Chong Zhang, Qi Zhang, and Zhongyu Wei. 2021b. [Learning implicit sentiment in aspect-based sentiment analysis with supervised contrastive pre-training](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 246–256. Association for Computational Linguistics.
- Qiao Liu, Haibin Zhang, Yifu Zeng, Ziqi Huang, and Zufeng Wu. 2018. [Content attention model for aspect based sentiment analysis](#). In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1023–1032. ACM.
- 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*, pages 3538–3547. Association for Computational Linguistics.
- 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*, pages 4068–4074. ijcai.org.
- Rajdeep Mukherjee, Shreyas Shetty, Subrata Chattopadhyay, Subhadeep Maji, Samik Datta, and Pawan Goyal. 2021. [Reproducibility, replicability and beyond: Assessing production readiness of aspect based sentiment analysis in the wild](#). In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, volume 12657 of *Lecture Notes in Computer Science*, pages 92–106. Springer.
- Haiyun Peng, Yukun Ma, Yang Li, and Erik Cambria. 2018. [Learning multi-grained aspect target sequence for chinese sentiment analysis](#). *Knowl. Based Syst.*, 148:167–176.
- 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*, pages 3211–3220. Association for Computational Linguistics.
- 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*, pages 19–30. The Association for Computer Linguistics.
- 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*, pages 486–495. The Association for Computer Linguistics.
- 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*, pages 27–35. The Association for Computer Linguistics.
- Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. [Targeted sentiment classification with attentional encoder network](#). In *Artificial Neural Networks and Machine Learning - ICANN 2019: Text and Time Series - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part IV*, volume 11730 of *Lecture Notes in Computer Science*, pages 93–103. Springer.
- Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. [Effective lstms for target-dependent sentiment classification](#). In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 3298–3307. ACL.
- Duyu Tang, Bing Qin, and Ting Liu. 2016b. [Aspect level sentiment classification with deep memory network](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 214–224. The Association for Computational Linguistics.
- 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*, pages 2910–2922. Association for Computational Linguistics.

- 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*, pages 3002–3012. Association for Computational Linguistics.
- Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. [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*, pages 616–626. The Association for Computational Linguistics.
- 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*, pages 3316–3322. AAAI Press.
- Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016b. [Attention-based LSTM for aspect-level sentiment classification](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 606–615. The Association for Computational Linguistics.
- Xiaoyu Xing, Zhijing Jin, Di Jin, Bingning Wang, Qi Zhang, and Xuanjing Huang. 2020. [Tasty burgers, soggy fries: Probing aspect robustness in aspect-based sentiment analysis](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3594–3605. Association for Computational Linguistics.
- 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*, pages 592–598. Association for Computational Linguistics.
- Mayi Xu, Biqing Zeng, Heng Yang, Junlong Chi, Jiatao Chen, and Hongye Liu. 2022. [Combining dynamic local context focus and dependency cluster attention for aspect-level sentiment classification](#). *Neurocomputing*, 478:49–69.
- Heng Yang. 2019. [PyABSA: Open Framework for Aspect-based Sentiment Analysis](#).
- Heng Yang and Ke Li. 2022. [Augmentor or filter? reconsider the role of pre-trained language model in text classification augmentation](#). *CoRR*, abs/2210.02941.
- Heng Yang and Biqing Zeng. 2020. [Enhancing fine-grained sentiment classification exploiting local context embedding](#). *CoRR*, abs/2010.00767.
- Heng Yang, Biqing Zeng, Mayi Xu, and Tianxing Wang. 2021a. [Back to reality: Leveraging pattern-driven modeling to enable affordable sentiment dependency learning](#). *CoRR*, abs/2110.08604.
- Heng Yang, Biqing Zeng, Jianhao Yang, Youwei Song, and Ruyang Xu. 2021b. [A multi-task learning model for chinese-oriented aspect polarity classification and aspect term extraction](#). *Neurocomputing*, 419:344–356.
- 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*, pages 844–855. International Committee on Computational Linguistics.
- 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*, pages 2979–2985. IJCAI/AAAI Press.
- Biqing Zeng, Heng Yang, Ruyang Xu, Wu Zhou, and Xuli Han. 2019. [Lcf: A local context focus mechanism for aspect-based sentiment classification](#). *Applied Sciences*, 9(16):3389.
- 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*, pages 4567–4577. Association for Computational Linguistics.
- 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, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 6736–6745. International Committee on Computational Linguistics.
- Pinlong Zhao, Linlin Hou, and Ou Wu. 2020. [Modeling sentiment dependencies with graph convolutional networks for aspect-level sentiment classification](#). *Knowl. Based Syst.*, 193:105443.

A Related Works

In recent years, many open-source models have been developed for aspect-based sentiment classification (ASC) (Li et al., 2021a; Tian et al.,

2021; Li et al., 2021b; Wang et al., 2021) and aspect term extraction and sentiment classification (ATESC) (Li et al., 2018b; Xu et al., 2018; Ma et al., 2019; Yang, 2019; Yang et al., 2020). However, the open-source repositories for these models often lack the capability to make predictions, and many are no longer being maintained. Two works similar to PyABSA are ABSA-PyTorch (Song et al., 2019) and Aspect-based Sentiment Analysis. ABSA-PyTorch combined multiple third-party models to facilitate fair comparisons of accuracy and F1, but it is now outdated and only supports the ASC task. Aspect-based Sentiment Analysis (Consultants, 2020) also handles ASC, but with limited models. PyABSA is a research-friendly framework that supports multiple aspect-based sentiment analysis (ABSA) subtasks and includes multilingual, open-source ABSA datasets. The framework has instant inference interfaces for both aspect-based sentiment classification (ASC) and aspect-term extraction and sentiment classification (ATESC) subtasks, facilitating the implementation of multilingual ABSA services. PyABSA sets itself apart from other similar works, such as ABSA-PyTorch and Aspect-based Sentiment Analysis, by being actively maintained and supporting multiple ABSA subtasks.

B Model Evaluation

We present the experimental results of various models on different datasets, which may help users choose a suitable model for their projects.

C Metric Visualization in PyABSA

C.1 Code for Auto-metric Visualization

PyABSA provides standardised methods for monitoring metrics and metric visualisations. PyABSA will automatically generate trajectory plot, box plot, violin plot, and bar charts based on metrics to evaluate the performance differences across models, etc. This example aims at evaluating the influence of maximum modelling length as a hyperparameter on the performance of the FAST-LSA-T-V2 model on the Laptop14 dataset.

```
import random
import os
from metric_visualizer import MetricVisualizer

from pyabsa import AspectSentimentClassification as ASC

config = ASC.ASCConfigManager.get_config_english()
config.model = ASC.ASCModelList.FAST_LSA_T_V2
config.lcf = 'cdw'

# each trial repeats with different seed
config.seed = [random.randint(0, 10000) for _ in range(3)]
```

Table 3: The evaluation of the performance of the ASC and ATESC models on the datasets available in `PYABSA`. The results in parentheses are the standard deviations. These results were obtained through 10 epochs of training using the default settings. The multi-language dataset includes all the built-in datasets from `PYABSA`. The absence of results for some datasets using syntax-based models is indicated by "-".

[illegible]

```

MV = MetricVisualizer()
config.MV = MV

max_seq_lens = [50, 60, 70, 80, 90]

for max_seq_len in max_seq_lens:
    config.max_seq_len = max_seq_len
    dataset = ABSADatasetList.Laptop14
    Trainer(config=config,
            dataset=dataset,
            auto_device=True
            )
    config.MV.next_trial()

save_prefix = os.getcwd()
# save fig into .tex and .pdf format
MV.summary(save_path=save_prefix)

MV.to_excel(save_path=os.getcwd() + "/example.xlsx") # save
to excel
MV.box_plot(no_overlap=True, save_path="box_plot.png")
MV.violin_plot(no_overlap=True, save_path="violin_plot.png")
MV.scatter_plot(save_path="scatter_plot.png")
MV.trajectory_plot(save_path="trajectory_plot.png")

MV.scott_knott_plot()
MV.A12_bar_plot()

```

C.2 Automatic Metric Visualizations

There are some visualization examples auto-generated by PyABSA. Note that the metrics are not stable on small datasets.

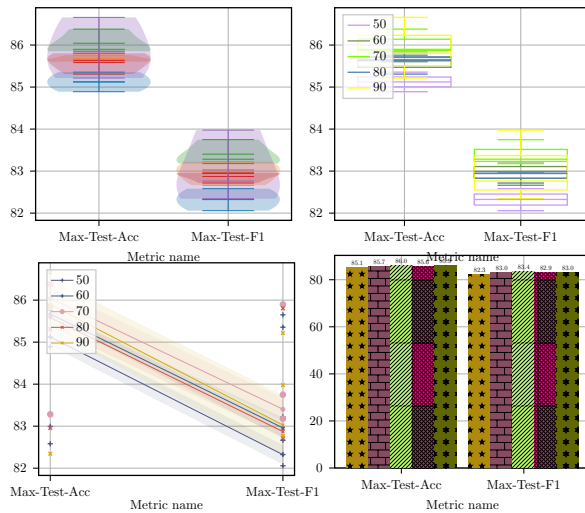


Figure 5: An example of automated -metric visualizations of the Fast-LSA-T-V2 model grouped by metric names.

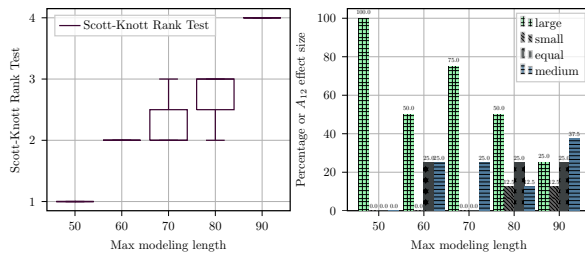


Figure 6: The significance level visualizations of the Fast-LSA-T-V2 grouped by different max modeling length. The left is scott-knott rank test plot, while the right is A12 effect size plot.