PyABSA: Open Framework for Aspect-based Sentiment Analysis

Preprint · August 2022

DOI: 10.48550/arXiv.2208.01368

Heng Yang
University of Exeter

PyABSA: Open Framework for Aspect-based Sentiment Analysis

Heng Yang[1], **Ke Li**[1]
[1]Department of Computer Science, University of Exeter, EX4 4QF, Exeter, UK
{hy345,k.li}@exeter.ac.uk

Abstract

Aspect-based sentiment analysis (ABSA) has become a prevalent task in recent years. However, the absence of a unified framework in the present ABSA research makes it challenging to compare different models' performance fairly. Therefore, we develop an open-source ABSA framework, namely PyABSA. Besides, previous efforts usually neglect the precursor aspect term extraction (ASC) subtask and focus on the aspect sentiment classification (ATE) subtask, while PyABSA includes the features of aspect term extraction, aspect sentiment classification, and text classification. Furthermore, multiple ABSA subtasks can be adapted to PyABSA owing to its modular architecture. To facilitate ABSA applications, PyABSAseamless integrates multilingual modelling, automated dataset annotation, etc., which are helpful in deploying ABSA services. In ASC and ATE, PyABSA provides up to 33 and 7 built-in models, respectively, while all the models provide quick training and instant inference. Besides, PyABSA contains to $180K$+ ABSA examples from 21 augmented ABSA datasets for applications and studies. PyABSA is available at https://github.com/yangheng95/PyABSA.

1 Introduction

Recent years, aspect-based sentiment analysis (Pontiki et al., 2014, 2015, 2016) have seen a tremendous improvement, particularly the subtasks of 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 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). The various open-source models' architectures (e.g., BERT or LSTM) and implementation details (e.g., data pre-processing methods) are diverse, making performance comparisons and experimental results reproduction challenging. On the other hand, even though ABSA is an application-driven task, earlier works tend to release research-oriented code without promising any inference support. These models are unavailable in practical applications because they are difficult to deploy. Therefore, in order to reduce the model performance deviation casused by model-irrelevant code and encourage fair comparisons, we introduce an open-source unified framework for aspect-based sentiment analysis that supports quick model training, evaluation, and inference. With only a few lines of code, everyone can train and deploy a model using PyABSA's user-friendly interfaces[1].

Although, previous works generally focus on the ASC[2] subtask, aspect-based sentiment analysis includes many other subtasks. For example, aspect category detection (ACD). PyABSA provides the multi-task based ATESC models, which is a pipeline model that simultaneously performs ATE and ASC subtasks. We build PyABSA into five main modules: dataset manager, data preprocessor, hyperparameter manager, trainer, and checkpoint manager. In this design, PyABSA allows the addition of other subtasks or models based on provided templates. Furthermore, the modular structure makes it easy to deploy an ABSA service in a Python environment. PyABSA offers more than 40 models (including variants), with some of them achieving state-of-the-art performance according to Paperswithcode[3], making it one of the most accessible and user-friendly ABSA frameworks.

When compared to other NLP tasks, ABSA suffers a more significant data shortage, which leads to problems such as performance volatility and

[1]We provides an ATESC inference service demo on https://huggingface.co/spaces/yangheng/Multilingual-Aspect-Based-Sentiment-Analysis

[2]We refer to ASC as aspect polarity classification (APC) in PyABSA's implementation.

[3]Our Fast-LSA model and LCF-ATEPC model achieve state-of-the-art performance on ASC subtask.

limited domain coverage. Hence, we provide automated dataset annotation interface and manual dataset annotation tool to encourage the community to annotate and contribute custom datasets to PyABSA [4] to tackle the data shortage problem. Currently, PyABSA provides up to 21 ABSA datasets in 8 languages that cover several domains. To the best of our knowledge, PyABSA offers the most open source datasets compared to other open-source projects. In addition, based on our self-developed ABSA data augmentor[5], we are able to provide up to $180K+$ ABSA examples. According to our evaluation, the augmented datasets can boost performance by $1-3\%$. Inspired by the Transformers[6], we pre-trained a range of models using these datasets and released checkpoints[7] that are available for users to fine-tune with custom datasets to increase model robustness and performance, especially for multilingual data.

Compare to previous works, PyABSA contains the following features:

- PyABSA provides quick training and fair evaluation for all models, which can alleviate the deviations in different model comparisons.
- PyABSA contains enough multilingual built-in datasets from different domains for users to study. Anyone can annotate a custom dataset based on PyABSA and contribute the custom dataset to PyABSA to enrich the open-source ABSA datasets.
- PyABSA is equipped with some user-friendly features compared to other works. e.g., model ensembling, dataset ensembling, and auto-metric visualization.

2 Frame Architecture

Figure 1 demonstrates the main architecture of PyABSA. As an extensible ABSA framework, PyABSA is decomposed into five core modules to accommodate a variety of subtasks or models. In the following sections, we will discuss the primary modules.

[4] We release the integrated datasets at https://github.com/yangheng95/ABSADatasets. All the public and community-shared datasets are released under its own license.

[5] This ABSA dataset augmentor will be open-sourced soon.

[6] https://github.com/huggingface/transformers

[7] https://huggingface.co/spaces/yangheng/Multilingual-Aspect-Based-Sentiment-Analysis/blob/main/checkpoint-v1.16.json

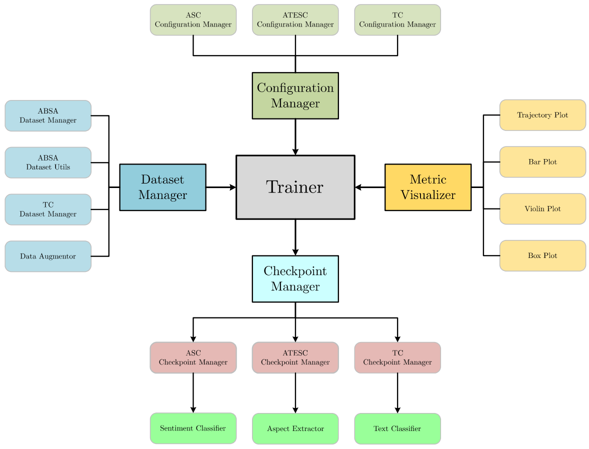


Figure 1: The main architecture of PyABSA framework.

2.1 Configuration Manager

Each ABSA subtask has an independent configuration manager used for training environment configuration, model hyperparameter configuration and other global configurations. The configuration manager extends the Python Namespace object to improve usability. We also set a hyperparameter calling counter to help users check training setting and debug code. The configuration manager applies a hyperparameter check module with the purpose of terminating execution if any hyperparameter is incorrectly specified. PyABSA saves and loads the configuration manager object synchronously with the checkpoints and prints all configuration information and calling counters during training and inference.

2.2 Trainer

The trainer is used for ABSA and text classification model training. The PyABSA trainer assembles a set of models and datasets based on the configuration manager's settings. The trainer supports k-fold cross-validation and accepts model output as a Python dictionary object, enabling users to utilise custom loss within the model and improve the model's ability to learn features with custom loss functions without modifying the framework. When there are insufficient datasets, it is often required to combine the training and testing sets to obtain a ensemble dataset for training. This is due to the fact that the trainer automatically detects the existence of the testing and validation sets, and if neither exists, it continues training without validation rather than cancelling the training.

2.3 Dataset Manager

The dataset manager is utilised to manage built-in and custom datasets. In PyABSA, each dataset object has a unique ID, name, and data file list. All the datasets are easy to combine for ensemble training and testing. The Dataset manager is responsible for handling automatic dataset downloading and datafile location. In particular, we include both automated and manual dataset annotation methods in ABSA Dataset Utils, enabling users to use PyABSA for custom datasets compared to existing work. This is a really effective way to annotate custom datasets. In the case of community-contributed Yelp datasets, the built-in data augmentation method can improve performance by 2% to 4% (refer to Section 4.3). Before feeding the data into the model, the dataset manager delivers the loaded data to the suitable data processor of the model for further processing.

2.4 Checkpoint Manager

PyABSA instantiates the inference models using the checkpoint manager, which provides the following three ways for loading checkpoints, in order to standardise the inference process across distinct subtasks.

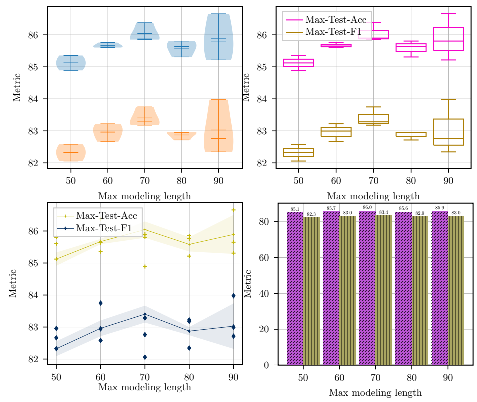
- Querying available cloud checkpoints and downloading them automatically by name.
- Using keywords or paths, the checkpoint manager can search for local checkpoints.
- Acquiring trained models return by the trainers to build inference models.

The checkpoint manager for any subtask is compatible with GloVe and pre-trained models based on transformers. Using the interface provided by PyABSA, everyone just need a few lines of code when launching an ATESC service (refer to Section 4.2).

2.5 Metric Visualizer

As a vital step toward facilitating accurate evaluation and fair comparison, we developed the metric visualizer[8] to automatically record, measure, and visualize a wide variety of metrics (such as accuracy, F-measure, standard deviation, IQR, etc.) for various models. The metric visualizer can monitor metrics or load metrics records to create many plots (e.g., box plots, violin plots, trajectory plots, Scott-Knott test plots, etc.). Figure 2 show an example of auto-visualizations generated by the metric visualizer, refer to Appendix 7.2.1 for more plots and related experiment setting. The metric visualizer reduces the difficulty of visualizing performance metrics and eliminates metric statistical biases.

Figure 2: An example of auto-metric visualizations of the Fast-LSA-T-V2 model grouped by maximum modeling length.



3 Models & Dataset

3.1 Models

PyABSA provides a range of models developed for ABSA, including LSA models for aspect sentiment classification and LCF-ATESC models for aspect term extraction. We also merge some popular ASC models from ABSA-PyTorch in order to provide users with more choices. The models currently supported by PyABSA are shown in Table 1, and we provide users with templates for creating their own models.

3.2 Datasets

PyABSA contains datasets on laptops, restaurants, MOOCs, Twitter, and other domains in 8 languages. As far as we are aware, PyABSA is one of the repositories with the most open-source datasets. The available ABSA datasets can be found in Table 3. Based on the built-in datasets, anyone can conduct their own research or augment their own datasets by merging built-in datasets in training.

3.3 Performance Evaluation

We have conducted some preliminary performance evaluations based on the models and datasets provided by PyABSA to help users understand the performance variations between the various models; the experimental results are available in Table 2.

[8]We develop the metric visualizer for PyABSA, it is released as an independent open-source project for other usages at: https://github.com/yangheng95/metric-visualizer

Table 1: The prevalent built-in models provided by PyABSA. Note that the models based on BERT can be adapted to other pre-trained models from HuggingFace Transformers. "LCF" indicates our own models based on the local context focus mechanism.

Model	Task	Paper	GloVe	PTM	LCF	Easy Inference
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-BASE-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	ATESC	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)	✗	✓	✓	✓
LSTM-TC	TC	Hochreiter et al. (1997)	✓	✗	✗	✓
BERT-BASE-TC		Devlin et al. (2019)	✗	✓	✗	✓

Note that we adopt the default hyperparameter configurations for each model. Altering these hyperparameters may result in significant performance variations.

4 Quick Examples

PyABSA is an application-oriented framework that provides brief interfaces to allow unified training and inference. With PyABSA, users are able to run training and inference with only a few lines of code. Section 4.1 and Section 4.2 show the quick training and inference examples based on PyABSA. Furthermore, PyABSA is an available package on PyPi[9], allowing users to integrate it into any existing application as a dependent.

4.1 Training

The following code snippet demonstrates an example of quick ATESC training on a multilingual dataset using the Fast-LCF-ATESC model.

```
from pyabsa.functional import (ATEPCConfigManager,
                               Trainer,
                               ABSADatasetList)

config = ATEPCConfigManager.get_atepc_config_multilingual()
multilingual = ABSADatasetList.Multilingual
aspect_extractor = Trainer(config=config,
                           dataset=multilingual,
                           ).load_trained_model()
```

[9]https://pypi.org/project/PyABSA

Table 2: Performance evaluation of the ASC and ATESC models on the datasets provided by PyABSA. The experimental results in parentheses are standard deviations. All the experimental results are obtained by ten epochs of training using the default configurations. The multilingual dataset is composed of all the built-in datasets from PyABSA. "–" indicates that the syntax-based models are unavailable for target datasets.

ASC Model	Task	English		Chinese		Arabic		Dutch		Spanish		French		Turkish		Russian		Multilingual	
		Acc_{ASC}	$F1_{ASC}$	Acc_{ASC}	$F1_{ASC}$	Acc_{ASC}	$F1_{ASC}$	Acc_{ASC}	$F1_{ASC}$	Acc_{ASC}	$F1_{ASC}$	Acc_{ASC}	$F1_{ASC}$	Acc_{ASC}	$F1_{ASC}$	Acc_{ASC}	$F1_{ASC}$	Acc_{ASC}	$F1_{ASC}$
BERT-SPC	ASC	84.57(0.44)	81.98(0.22)	95.27(0.29)	94.10(0.40)	95.27(0.29)	94.10(0.40)	89.12(0.21)	74.64(0.68)	86.29(0.51)	68.64(0.26)	86.14(0.63)	75.59(0.45)	85.27(0.34)	65.58(0.07)	87.62(77.13)	77.13(0.08)	87.19(0.36)	80.93(0.21)
DLCF-DCA		84.05(0.06)	81.03(1.05)	95.69(0.22)	94.74(0.30)	89.23(0.15)	75.68(0.01)	86.93(0.38)	72.81(1.57)	86.42(0.49)	76.29(0.10)	90.42(0.0)	73.69(0.82)	85.96(1.71)	67.59(1.61)	87.00(0.41)	74.88(0.41)	87.80(0.01)	81.62(0.20)
Fast-LCF-ASC		84.70(0.05)	82.00(0.08)	95.98(0.02)	95.01(0.05)	89.82(0.06)	77.68(0.33)	86.42(0.13)	71.36(0.53)	86.35(0.28)	75.10(0.14)	91.59(0.21)	72.31(0.26)	86.64(1.71)	67.00(1.63)	87.77(0.15)	74.17(0.47)	87.66(0.15)	81.34(0.25)
Fast-LCFS-ASC		84.27(0.09)	81.60(0.17)	95.67(0.32)	94.40(0.55)	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LCF-BERT		84.81(0.29)	82.06(0.06)	96.30(0.05)	95.45(0.05)	89.80(0.13)	77.60(0.44)	86.55(0.76)	70.67(0.41)	85.52(0.42)	74.03(0.75)	91.86(0.21)	75.26(0.37)	89.73(0.05)	68.57(1.09)	87.41(0.21)	74.71(0.17)	87.86(0.09)	82.01(0.46)
LCFS-BERT		84.49(0.13)	81.46(0.05)	95.32(0.39)	94.23(0.56)	88.89(0.11)	75.41(0.37)	87.94(0.43)	72.69(1.01)	84.61(0.21)	71.98(1.25)	90.83(0.41)	73.87(1.45)	88.36(0.68)	69.21(0.86)	87.15(0.15)	74.99(0.44)	87.55(0.22)	81.58(0.13)
Fast-LSA-T		84.60(0.29)	81.77(0.44)	96.05(0.05)	95.10(0.05)	89.25(0.38)	77.25(0.43)	86.04(0.0)	70.02(0.75)	86.07(0.14)	73.52(0.53)	91.93(0.27)	74.21(0.60)	88.01(1.03)	66.74(0.61)	88.24(0.10)	76.91(1.10)	87.56(0.13)	81.01(0.56)
Fast-LSA-S		84.15(0.15)	81.53(0.03)	89.55(0.11)	75.87(0.25)	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Fast-LSA-P		84.21(0.06)	81.60(0.23)	95.27(0.29)	94.10(0.40)	89.12(0.21)	74.64(0.68)	86.29(0.51)	68.64(0.26)	86.14(0.63)	75.59(0.45)	90.77(0.07)	74.60(0.82)	85.27(0.34)	65.58(0.07)	87.62(0.06)	77.13(0.08)	87.81(0.24)	81.58(0.56)
ATESC Model	**Task**	**English**		**Chinese**		**Arabic**		**Dutch**		**Spanish**		**French**		**Turkish**		**Russian**		**Multilingual**	
		$F1_{ASC}$	$F1_{ATE}$	$F1_{ASC}$	$F1_{ATE}$	$F1_{ASC}$	$F1_{ATE}$	$F1_{ASC}$	$F1_{ATE}$	$F1_{ASC}$	$F1_{ATE}$	$F1_{ASC}$	$F1_{ATE}$	$F1_{ASC}$	$F1_{ATE}$	$F1_{ASC}$	$F1_{ATE}$	$F1_{ASC}$	$F1_{ATE}$
BERT-ATESC	ATESC	72.70(0.48)	81.66(1.16)	94.12(0.12)	64.86(0.50)	71.18(0.34)	71.18(0.34)	71.06(1.09)	78.31(1.40)	69.29(0.07)	83.54(0.59)	69.26(0.07)	83.54(0.59)	67.28(0.37)	72.88(0.37)	70.46(0.54)	77.64(0.19)	75.13(0.15)	80.9(0.02)
Fast-LCF-ASESC		79.23(0.07)	81.78(0.12)	94.32(0.29)	84.15(0.39)	67.38(0.11)	70.30(0.41)	73.67(0.89)	78.59(0.69)	71.24(0.47)	83.37(1.21)	71.24(0.47)	82.06(0.67)	67.64(1.28)	73.46(0.87)	71.28(0.37)	76.90(0.52)	78.96(0.13)	80.31(0.19)
Fast-LCFS-ASESC		75.82(0.03)	81.40(0.59)	93.68(0.25)	84.48(0.32)	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LCF-ATESC		77.91(0.41)	82.34(0.35)	94.00(0.38)	84.64(0.38)	67.30(0.17)	70.52(0.28)	71.85(1.53)	79.94(1.70)	70.19(0.24)	84.22(0.83)	70.76(0.92)	82.16(0.38)	69.52(0.54)	74.88(0.08)	71.96(0.28)	79.06(0.52)	80.63(0.35)	80.15(1.18)
LCFS-ATESC		75.85(0.22)	85.00(1.58)	93.52(0.09)	84.92(0.11)	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 3: The details of datasets in different languages available in PYABSA, where "†" indicates the datasets are used for adversarial attack study. The datasets denoted by "*" are a subset with $10K$ examples from the original datasets. The augmented examples of the training set are generated by our own ABSA augmentor.

Dataset	Task	Language	# of Examples			# of Augmented Examples	Source
			Training Set	Validation Set	Testing Set	Training Set	
Laptop14	ASC / ATESC	English	2328	0	638	13325	SemEval 2014
Restaurant14	ASC / ATESC	English	3604	0	1120	19832	SemEval 2014
Restaurant15	ASC / ATESC	English	1200	0	539	7311	SemEval 2015
Restaurant16	ASC / ATESC	English	1744	0	614	10372	SemEval 2016
Twitter	ASC / ATESC	English	5880	0	654	35227	Dong et al. (2014)
MAMS	ASC / ATESC	English	11181	1332	1336	62665	Jiang et al. (2019)
Television	ASC / ATESC	English	3647	0	915	25676	Mukherjee et al. (2021)
T-shirt	ASC / ATESC	English	1834	0	465	15086	Mukherjee et al. (2021)
Yelp	ASC / ATESC	English	808	0	245	2547	WeiLi9811@GitHub
Phone	ASC / ATESC	Chinese	1740	0	647	0	Peng et al. (2018)
Car	ASC / ATESC	Chinese	862	0	284	0	Peng et al. (2018)
Notebook	ASC / ATESC	Chinese	464	0	154	0	Peng et al. (2018)
Camera	ASC / ATESC	Chinese	1500	0	571	0	Peng et al. (2018)
MOOC	ASC / ATESC	Chinese	1583	0	396	0	jmc-123@GitHub
Shampoo	ASC / ATESC	Chinese	6810	0	915	0	brightgems@GitHub
MOOC-En	ASC / ATESC	English	1492	0	459	10562	aparnavalli@GitHub
Arabic	ASC / ATESC	Arabic	9620	0	2372	0	SemEval 2016
Dutch	ASC / ATESC	Dutch	1283	0	394	0	SemEval 2016
Spanish	ASC / ATESC	Spanish	1928	0	731	0	SemEval 2016
Turkish	ASC / ATESC	Turkish	1385	0	146	0	SemEval 2016
Russian	ASC / ATESC	Russian	3157	0	969	0	SemEval 2016
French	ASC / ATESC	French	1769	0	718	0	SemEval 2016
ARTS-Laptop14†	ASC / ATESC	English	2328	638	1877	13325	Xing et al. (2020)
ARTS-Laptop15†	ASC / ATESC	English	3604	1120	3448	19832	Xing et al. (2020)
SST2	TC	English	6920	872	1821	0	Socher et al. (2013)
SST5	TC	English	8544	1101	2210	0	Stanford Sentiment Treebank
AGNews*	TC	English	7000	1000	2000	0	Zhang et al. (2015)
IMDB*	TC	English	7000	1000	2000	0	Maas et al. (2011)
Yelp*	TC	English	7000	1000	2000	0	Zhang et al. (2015)

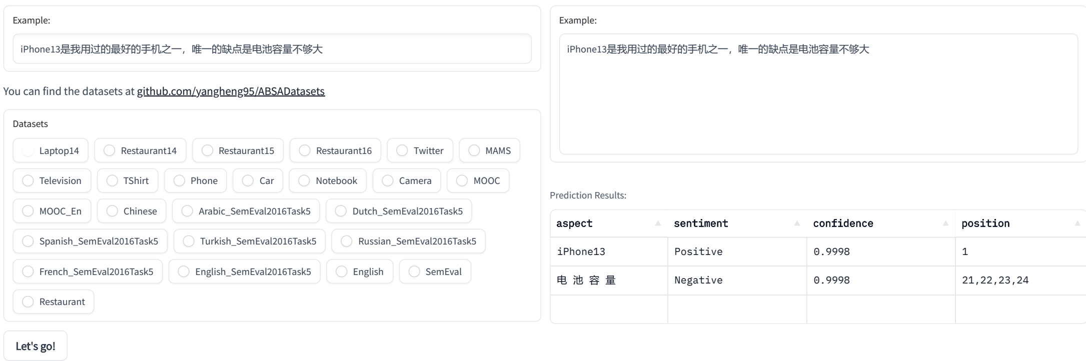


Figure 3: An example from the demo of multilingual aspect term extraction and sentiment classification. Our demo accepts user input or random example from existing datasets.

4.2 Inference

PYABSA supports quick training and inference for all models, regardless of whether they are based on Word2Vec or a pre-trained language model. Figure 3 shows an inference example output by the demo ATESC service we launched on the Huggingface Space. Section 4.1 indicates different ways to obtain a inference model:

```
from pyabsa import ATEPCCheckpointManager,
      available_checkpoints, ABSADatasetList

available_checkpoint = available_checkpoints()
AspectExtractor = ATEPCCheckpointManager.
      get_aspect_extractor(checkpoint='multilingual')
# Load a local checkpoint by specifying the checkpoint path.
# AspectExtractor = ATEPCCheckpointManager.
      get_aspect_extractor(checkpoint='./checkpoints/
      multilingual')

examples = ["But the staff was so nice to us ."]
atepc_result = aspect_extractor.extract_aspect(
      inference_source=examples, pred_sentiment=True)
```

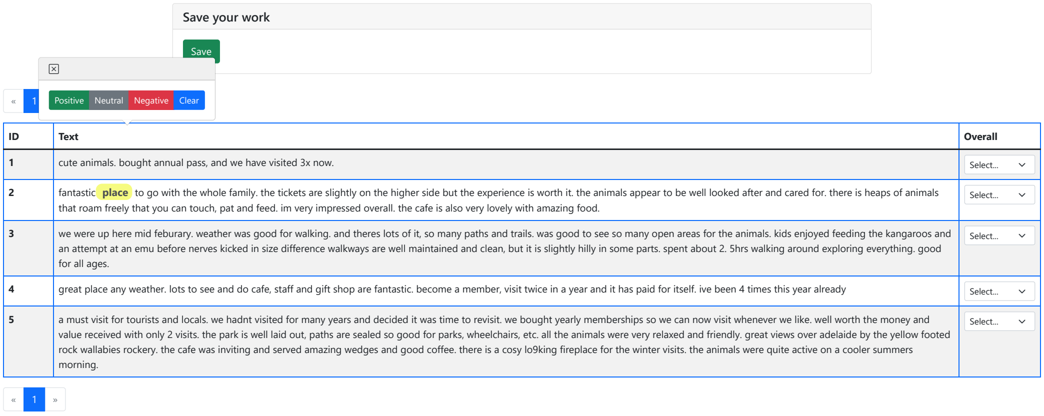
4.3 Dataset Annotation

There is currently no open-source tool for annotating ABSA datasets, making the creation of custom datasets very difficult. In PYABSA, both an automatic annotated interface and a manual tool for dataset annotation are included. Using our automated interface for dataset annotation, users may quickly complete the annotation, training, and deployment of inference models.

4.3.1 Automated Annotation

Unlike text classification, annotating ASC and ATESC datasets is difficult. Therefore, we developed an automated dataset annotation interface based on the ATESC inference model, converting ATESC inference results into ASC and ATESC

Figure 4: The community contributed manual dataset annotation tool provided for PyABSA.



annotations. Note that the $F1$ measures on the Chinese, English, and multilingual datasets are up to $\sim 85\%$. In this case,this method is helpful for obtaining annotated ABSA instances to augment small datasets. The following example illustrates the interface of automated dataset annotation.

```
from pyabsa import make_ABSA_dataset

make_ABSA_dataset(dataset_name_or_path='raw_data',
      checkpoint='multilingual')
```

4.3.2 Manual Annotation

In order to conduct precise manual annotation, our contributor developed a specialized ASC annotation tool[10] for PyABSA. Furthermore, we provide an interface for converting existing ASC datasets to ATESC datasets. Figure 4 shows the manual annotation tool.

5 Related Works

In recent years, a large number of outstanding open-source ASC (Li et al., 2021a; Tian et al., 2021; Li et al., 2021b; Wang et al., 2021) and ATESC (Li et al., 2018b; Xu et al., 2018; Ma et al., 2019; Yang, 2019; Yang et al., 2020) models have been proposed. However, the related open-source repositories for these models usually lack inference support, and most of them are no longer maintained. There are two works most similar to PyABSA, ABSA-PyTorch and Aspect-based Sentiment Analysis, respectively. ABSA-PyTorch (Song et al., 2019) incorporates multiple reimplemented third-party GloVe-based and BERT-based models as an early effort to propagate fair comparisons of accuracy and F1 amongst models. Nevertheless, ABSA-PyTorch is no longer maintained and only supports the ASC subtask. ASC subtasks are also handled by Aspect-based Sentiment Analysis (Consultants, 2020), which provides an ASC inference interface based on constrained models. PyABSA is a research- and application-friendly framework that supports a number of ABSA subtasks and includes multilingual, open-source ABSA datasets. We developed instant inference interfaces for ASC and ATESC subtasks, which facilitate the implementation of multilingual ABSA services, using inspiration from Transformers.

6 Conclusion

We developed an open-source ABSA framework, namely PyABSA. By diminishing the influence of model-irrelevant code and automating metric visualization, etc., PyABSA seeks to encourage fair comparisons across ABSA models. Furthermore, to facilitate ABSA applications, we implement instant ASC and ATESC inference interfaces that enable anyone to launch ABSA services with a few lines of code. For starters, PyABSA integrates a set of built-in models and datasets. It also encourages users to develop new models based on our templates or contribute custom datasets. PyABSA is a lightweight, open-source framework that can be added to any Python environment as a dependency to provide ASC and ATESC services. In the future, we plan to include more ABSA subtasks into PyABSA, such as aspect 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.

[10]https://github.com/yangheng95/ABSADatasets/DPT

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.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics.

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.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

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 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.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

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.

7 Appendix

7.1 Training and Inference Pipeline

Similar to ATESC, ASC may be completed with a few lines of code. We also provide a range of choices to make training easier for users; please see the function's comments for more information.

7.1.1 ASC Training Pipeline

```
from pyabsa.functional import Trainer
from pyabsa.functional import APCConfigManager
from pyabsa.functional import ABSADatasetList
from pyabsa.functional import APCModelList

apc_config_multilingual = APCConfigManager.
      get_apc_config_multilingual()
apc_config_multilingual.model = APCModelList.FAST_LSA_T_V2

datasets_path = ABSADatasetList.Multilingual
sent_classifier = Trainer(config=apc_config_multilingual,
                          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()
```

7.1.2 ASC Inference Example

```
from pyabsa import ABSADatasetList, APCCheckpointManager,
      available_checkpoints

checkpoint_map = available_checkpoints(from_local=False)

sent_classifier = APCCheckpointManager.
      get_sentiment_classifier(checkpoint='Multilingual')

text = 'everything is always cooked to perfection , the [ASP
      ]service[ASP] is excellent , the [ASP]decor[ASP] cool
      and understated . !sent! 1, 1'  # PyABSA indentifies
      targeted aspects warpped by '[ASP]' token
sent_classifier.infer(text, print_result=True)
```

7.1.3 Text Classification Training

```
from pyabsa import TCTrainer, TCConfigManager, TCDatasetList
config = TCConfigManager.get_tc_config_english()
config.cross_validate_fold = 5

dataset = TCDatasetList.SST2
text_classifier = TCTrainer(config=config,
                            dataset=dataset,
                            checkpoint_save_mode=1,
                            auto_device=True
                            ).load_trained_model()
```

7.1.4 Text Classification Inference Example

```
from pyabsa import TCDatasetList, TCCheckpointManager

model_path = 'lstm'  # 'lstm' is a keyword to search the
      checkpoint in the folder
text_classifier = TCCheckpointManager.get_text_classifier(
      checkpoint=model_path)

# batch inference works on the dataset files
inference_sets = TCDatasetList.SST2
results = text_classifier.batch_infer(target_file=
      inference_sets,
                                      print_result=True,
                                      save_result=True,
                                      ignore_error=True,
                                      )
```

7.2 Metric Visualization in PyABSA

7.2.1 Code for Auto-metric Visualization

PyABSA provides standardised methods for monitoring metrics and metric visualisations. PyASBA 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.functional import Trainer
from pyabsa.functional import APCConfigManager
from pyabsa.functional import ABSADatasetList
from pyabsa.functional import APCModelList

config = APCConfigManager.get_apc_config_english()
config.model = APCModelList.FAST_LSA_T_V2
config.lcf = 'cdw'

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

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, no_print=True)

# plots grouped by model name or setting name
MV.traj_plot_by_trial(save_path=save_prefix)
MV.violin_plot_by_trial(save_path=save_prefix)
MV.box_plot_by_trial(save_path=save_prefix)
MV.avg_bar_plot_by_trial(save_path=save_prefix)
MV.sum_bar_plot_by_trial(save_path=save_prefix)

# plots grouped by metric name
MV.traj_plot_by_trial(save_path=save_prefix)
MV.violin_plot_by_trial(save_path=save_prefix)
MV.box_plot_by_trial(save_path=save_prefix)
MV.avg_bar_plot_by_trial(save_path=save_prefix)
MV.sum_bar_plot_by_trial(save_path=save_prefix)

MV.scott_knott_plot(save_path=save_prefix)
MV.A12_bar_plot(save_path=save_prefix)
```

7.2.2 Visualizations

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

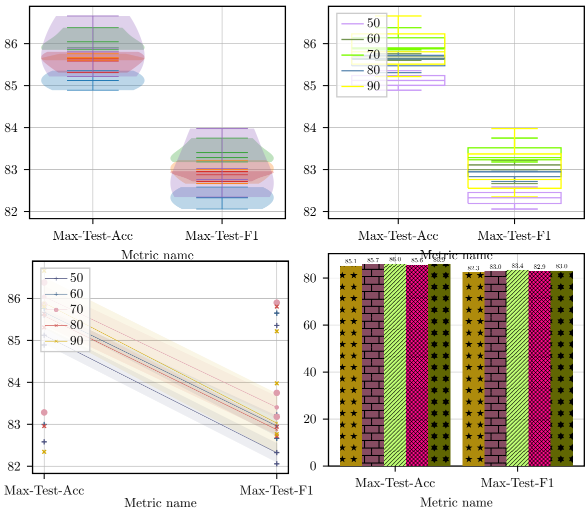


Figure 5: An example of auto-metric visualizations of the `Fast-LSA-T-V2` model grouped by metric names.

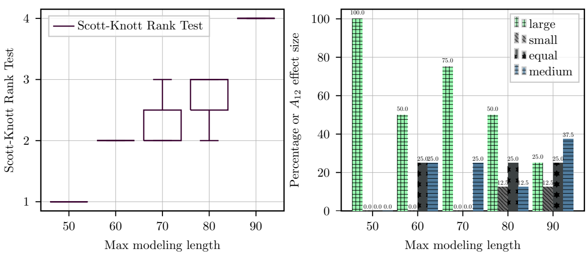


Figure 6: The significance level visualizations of the `Fast-LSA-T-V2` grouped by different max modeling length. The left is scott-knott rank test plot, while the right is A12 effect size plot.