# A Comprehensive Framework for Aspect-Category Sentiment Analysis

Loris Di Quilio[1], Fabio Fioravanti[1]

[1]DEc, University of Chieti-Pescara, Italy

**Abstract**

In this study, we developed an Aspect-Category Sentiment Analysis (ACSA) framework encompassing data conversion, semi-automatic annotation methods using predictions, and the creation of a prediction-based report. We aimed to adapt an Aspect-Category-Opinion Sentiment (ACOS) tool from the literature to the Aspect-Category Sentiment Analysis (ACSA) task. We developed a web application where the dataset released in this paper (beauty dataset) can be annotated manually or semi-automatically and incorporated into the training data to enhance the model. Additionally, we also evaluated our framework using various datasets available in the literature, comparing with a tool that follows a similar approach.

**Keywords**

Aspect-Category Sentiment Analysis (ACSA), Aspect-Based Sentiment Analysis (ABSA), Annotations, Sentiment Index

## 1. Introduction

Aspect category sentiment analysis (ACSA) is a sub-category of Aspect-Based Sentiment Analysis (ABSA) that aims at identifying the aspect categories and corresponding sentiments involved in a sentence, regardless of whether the aspect terms are explicitly mentioned or not [2].

This challenging task requires understanding context and linguistic nuances. For example, an aspect may be mentioned implicitly rather than explicitly, or a single piece of text may contain contrasting sentiments about different aspects [3]. ACSA provides detailed insights on various categories of products/services, helping in product development, marketing, and customer service. It automatizes the analysis of a large volume of customer feedbacks, identifying areas for improvement or differentiation, and informing strategic decisions and market responses.

In this paper we present *PyACSA*, a new ACSA tool that is based on a feature of *PyABSA*, used for the more complex task of Aspect Category Opinion Sentiment (ACOS) [4]. We developed a new dataset for the ACSA task on the Beauty and Personal Care domain. We created a framework with various functions: from converting datasets across various formats (SemEval2014[1], SemEval2015[2], SemEval2016[3], JSON), to manual and semi-automatic data annotation for this task, up to visualizing prediction data on graphs by calculating a sentiment index for each category of the domain. In addition, the PyACSA tool is experimented on several datasets available in the literature, demonstrating excellent results, comparing with *ACSA-Gen* [5], a state-of-the-art tool for ACSA.

## 2. The PyACSA tool

Aspect-Category Opinion Sentiment (ACOS) and Aspect-Category Sentiment Analysis (ACSA) are two Aspect-Based Sentiment Analysis (ABSA) tasks. They differ in that ACOS extracts four elements from the text (aspect terms, category, opinion terms, and sentiment polarity) [6, 7], whereas ACSA extracts

*NL4AI 2024: Eighth Workshop on Natural Language for Artificial Intelligence, November 26-27th, 2024, Bolzano, Italy [1]*
loris.diquilio@studenti.unich.it (L. Di Quilio); fabio.fioravanti@unich.it (F. Fioravanti)


[1]https://alt.qcri.org/semeval2014/task4/
[2]https://alt.qcri.org/semeval2015/task12/
[3]https://alt.qcri.org/semeval2016/task5/

two elements(category and sentiment polarity). In the table below we report an example that shows the extraction of all elements from a sentence.

| Text | Aspect Term | Category | Opinion Term | Sentiment polarity |
|---|---|---|---|---|
| *'Though the service might be a little slow, the waitresses are very friendly.'* | *'service'* | *'service'* | *'a little slow'* | *'negative'* |
| | *'waitresses'* | *'staff'* | *'very friendly'* | *'positive'* |

**Table 1**
Example of extraction of all ACOS elements from a sentence.

As mentioned in the introduction, we developed the PyACSA tool by specializing a new feature of PyABSA [4], which was designed for the more complex task of Aspect Category Opinion Sentiment, to the Aspect-Category Sentiment Analysis. It is important to highlight the fact that the task is carried with the same format as SemEval 2016 task 5, subtask 2 [8], which means at text-level. At text-level, given a customer review about a target entity, the goal is to identify a set of {*category*, *polarity*} pairs that summarize the opinions expressed in the review. The polarities can be "positive", "negative", "neutral" (when a category is mentioned without any sentiment), or "conflict" (when the same category is expressed in a positive and a negative inside the same text, but neither of the two is dominant).

The PyACSA tool uses the T5 (Text-to-Text Transfer Transformer) [9] model which is based on a standard encoder-decoder Transformer [10] capturing long-range dependencies in text. The T5 model, if trained on a vast corpus of text data, converts various natural language processing (NLP) tasks into a text-to-text format, achieving state-of-the-art performance on many benchmarks covering summarization, question answering, text classification, and more. The pre-trained model used in this work is `flan-t5-xl` [11], an extension of the original T5 model with improved performance.

The PyABSA tool, which is implemented in *PyTorch* [12], is built using this pre-trained model and fine-tuned on the dataset using several instructions, one for each element. We developed PyACSA by modifying the PyABSA code so that only one extraction is performed containing both categories and sentiment polarities. This method takes an input text, categories, and polarities, and returns a string that combines the instructions with the provided input. This formatted string is then fed to the model for training or prediction. A specific module facilitates the creation of a dataset for this task by preparing the data in the required format, creating the training and test datasets, and reading JSON data from a file. With this approach, we aim to simplify the implementation of the ACSA framework that exploits several utilities to facilitate this task. The tool's integration with PyABSA provides a robust and flexible system for our specific needs, allowing us to streamline the data preparation and model training processes exploring the application of a text-to-text model in a task where it is not typically used.

## 3. ACSA Utilities

In this section, we present the framework we developed for using and evaluating the PyACSA tool. We developed a web application in `Python` using the `Flask` package to promote the use of this task, facilitate user interaction with the model, and make the entire process more accessible and efficient. The framework contains key components such as data format transformation modules, for converting various input formats into the JSON format required by PyACSA), and utilities for manual and semi-automatic data annotation, allowing users to annotate data directly within the web app or use the model's predictions to assist in the annotation process. This dual approach enhances flexibility and efficiency, catering to different user needs and preferences, while leveraging the model for annotations improves accuracy and accelerates the workflow.

Furthermore, the framework contains a module that generates a bar chart along with a sentiment index (see Figure 3), which evaluates the categories of the reviews entered into the system. This feature provides valuable insights into the sentiment distribution across different aspects of our domain, aiding in the analysis and interpretation of the data. We release our code at

https://github.com/lorisdiquilio/ACSA-Framework-using-T2T-model.

In the following paragraphs, we describe the components of the framework in detail, highlighting their functionalities and expected benefits.

## 3.1. Data Converter

The data converter module provides various converters designed to facilitate the transformation of data across various formats, ensuring compatibility and ease of use for subsequent tasks. The module is composed of several functions written in `Python` to transform data across tasks, sub-tasks, and formats. Among the main converters we developed, there are those for conversion of data from SemEval 14, 15 and 16 (XML) format to the JSON format of PyACSA, which have been used to evaluate PyACSA on the SemEval datasets.

## 3.2. Data Annotation

This module allows the user to annotate the dataset in the JSON format used by our tool (shown in Listing 1). Data can be annotated both manually and in a semi-automatic way.

### 3.2.1. Manual annotation

This module allows to load a training file, in JSON format, that contains annotated text. After loading the file, it is possible to annotate each review, by adding new annotations selecting the categories and polarities, or by deleting annotations, if needed. Additionally, the module allows to add categories that are not already known to the system, offering flexibility and adaptability to various annotation needs.

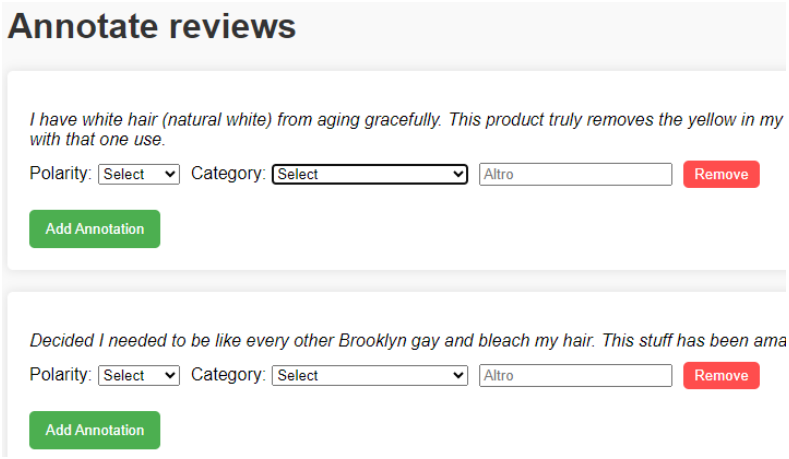


**Figure 1:** Manual annotation module

When the annotation process is finished, it is possible to export the annotations in JSON format.

### 3.2.2. Semi-automatic annotation

This module is designed to leverage the initial trained model for further improvements. Once a trained model is available, through this module we can use the model itself to suggest annotations on new data, making the dataset creation process faster and more efficient.

Similar to the manual annotation module, upon accessing this tab, the reviews and model predictions will be displayed, with the initial prediction next to the text. Multiple predictions for the same sentence will appear below the text. After reviewing the model's predictions, necessary corrections can be made and saved back into the original training file (JSON). This feature was used to annotate more data and improve model performance during experiments.

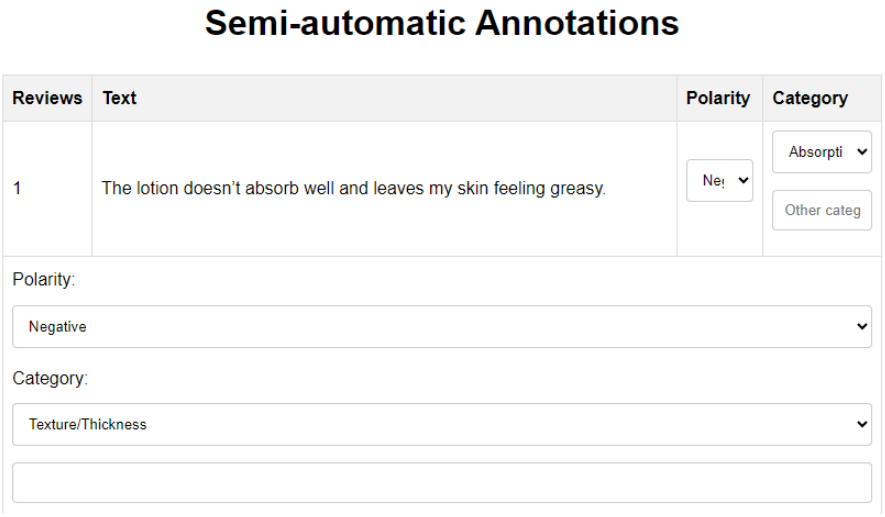


**Figure 2:** Semi-automatic annotation module. In this case the model predicts two tuples: {Absorption, Negative} and {Texture/Thickness, Negative}

.

In Figure 2 we see that the model predicts two correct tuples. It is easy to understand that in this way the annotation process becomes more efficient.

### 3.3. Report Generation

This module can be used to generate a report providing a final assessment of the reviews for a product.

We compute a sentiment index, which measures the overall quality of the product through all its reviews within the relevant domain. In this case, the domain represents a Skin Care, Body Care and Hair Care products. The sentiment index is computed based on the aggregated sentiment scores across different categories, offering a comprehensive evaluation of the product's performance. The sentiment index is computed as follows:

$$\text{Sentiment Index} = \frac{\text{Positive} - \text{Negative}}{\text{Positive} + \text{Negative}} \tag{1}$$

where Positive and Negative denote the number of positive and negative reviews, respectively. Note that the sentiment index value ranges from $-1$ to 1.

In Figure 3, we show the bar chart generated by the module, which displays the review polarities for each category of the selected product. This bar chart provides a visual representation of how users perceive different aspects of the product, with each bar indicating the level of positive or negative sentiment associated with a specific category. Figure 4 presents the overall sentiment index and the sentiment index calculated for each category. The overall sentiment index gives a comprehensive view of the general perception of the product, while the category-specific sentiment indices allow us to delve deeper into particular aspects. This dual representation helps in understanding not only the general acceptance of the product but also the specific areas where it excels or falls short.

From the results, we can evaluate the aspects that perform positively and negatively for this specific product. For instance, we can conclude that this product is generally well-received because it is effective, has a pleasant texture and smell, and is conveniently sized for travel. However, there are some drawbacks noted by users, such as the small quantity (for some), poor delivery and packaging, and the high cost of the product.

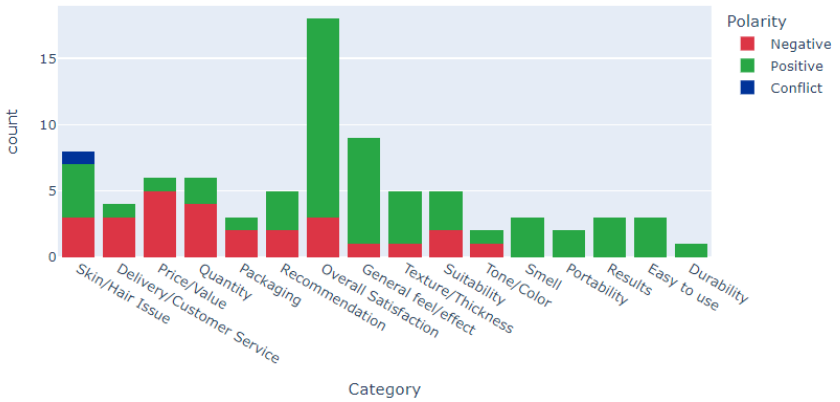


**Figure 3:** Bar chart showing review polarity per category

Overall Sentiment Index: 0.34

**Sentiment Index by Category:**

Delivery/Customer Service: -0.50
Durability: 1.00
Easy to use: 1.00
General feel/effect: 0.78
Overall Satisfaction: 0.67
Packaging: -0.33
Portability: 1.00
Price/Value: -0.67
Quantity: -0.33
Recommendation: 0.20
Results: 1.00
Skin/Hair Issue: 0.14
Smell: 1.00
Suitability: 0.20
Texture/Thickness: 0.60
Tone/Color: 0.00

**Figure 4:** Sentiment Indexes

## 4. Experimental evaluation

We have built an ACSA dataset based on Beauty and Personal Care reviews. The dataset was annotated manually and semi-automatically by one of the authors and is available at the repository.

In addition to the new dataset, we used some publicly available datasets[4]: SemEval Laptop and Restaurant, and MAMS (Multi-Aspect Multi-Sentiment). We used the Data conversion module of our framework to convert datasets in XML format to the JSON format used by PyACSA. In Table 2 we show some statistics about the datasets.

| | **Beauty** | MAMS | Rest 14 Hard | Rest 141516 Large-Hard | Rest 16 | Laptop |
|---|---|---|---|---|---|---|
| # Train | 1932 | 3549 | 137 | 270 | 335 | 395 |
| # Test | 218 | 400 | 25 | 48 | 90 | 80 |
| # Categories | 20 | 8 | 5 | 8 | 12 | 87 |
| # Positive annotations | 1124 | 2415 | 146 | 274 | 1298 | 1548 |
| # Negative annotations | 877 | 2606 | 143 | 259 | 411 | 870 |
| # Neutral annotations | 104 | 3858 | 59 | 168 | 78 | 154 |
| # Conflict annotations | 48 | - | - | - | 52 | 55 |

**Table 2**
Datasets used for the experimental evaluation

### 4.1. Experimental settings and results

In the experiment, the following settings are used: pre-trained `flan-t5-xl` with 3 billion parameters, learning rate of $5 \times 10^{-5}$. Epochs and batch size are set to 10 and 6, respectively. A regularization parameter ($L_2$ Regularization)[5] that helps prevent overfitting by reducing model weights during training is set to 0.01. The Warmup-Ratio (the ratio of total training steps used for a linear warmup from 0 to `learning_rate`) is set to 0.1.

The PyACSA tool is compared with ACSA-Gen [5], one of the state-of-the-art tools in this field, as stated in the survey [6], which also leverages the power of a pre-trained generative model. Note that

[4]https://github.com/l294265421/ACSA/tree/master/datasets
[5]https://paperswithcode.com/method/weight-decay

the model used by PyACSA is more powerful than that used by ACSA-Gen, which is a BART-large-MNLI[6], the best BART model for the classification task. The latter was used with the template "The sentiment polarity of <given_category> is <polarity_type>" for each label. The configurations for BART-large-MNLI are: learning rate of $4 \times 10^{-5}$, 15 epochs, and batch size of 16.

The results reported in the table are based on the most common metrics: Precision, Recall, and Micro-F1 Score.

| Tool | Metrics | Datasets | | | | | |
|---|---|---|---|---|---|---|---|
| | | Beauty | MAMS | Rest14 H | Rest141516 H | Rest16 | Lap16 |
| ACSA-Gen | P | 0.8109 | 0.7347 | 0.78 | 0.7333 | 0.8272 | 0.6567 |
| | R | 0.7477 | 0.7469 | 0.7358 | 0.7264 | 0.6163 | **0.3229** |
| | F1 | 0.7780 | 0.7407 | 0.7572 | 0.7298 | 0.7063 | 0.4329 |
| **PyACSA** | P | **0.8116** | **0.7741** | **0.8** | **0.8333** | **0.8316** | **0.6756** |
| | R | **0.8786** | **0.7913** | **0.7547** | **0.8018** | **0.8069** | 0.3211 |
| | F1 | **0.8438** | **0.7826** | **0.7766** | **0.8173** | **0.8190** | **0.4353** |

**Table 3**
Results of the experimental evaluation on different datasets

The results show that PyACSA performs better than ACSA-Gen in all the considered datasets, likely due to the power of the pre-trained model, as the BART model has a smaller size (approximately 406 million parameters) compared to the T5-XL (3 billion). We notice that PyACSA performs quite well in the ACSA task at the text level, except on the Laptop dataset, probably because of the high number of categories it contains.

## 5. Conclusion and future works

We adapted an existing tool from the literature to transition from an Aspect-Category-Opinion-Sentiment (ACOS) task to an Aspect-Category Sentiment Analysis (ACSA) task. We developed a comprehensive web application featuring various sections, including the transformation of different data formats for various Aspect-Based Sentiment Analysis tasks, using the model for data annotation, and creating a sentiment index to evaluate what are the performances in terms of topics in the reviews analyzed, entered into the tool. Additionally, we released a new dataset, which will be made available in the literature for future research in this domain, and we evaluate the tool by comparing it with a state-of-the-art tool. Our primary goal is to promote the use of this task by simplifying the entire underlying process, thereby facilitating broader adoption and application in the research community. However, there are some limitations to our current approach. One significant challenge is that loading the model is resource-intensive and requires a dedicated space for that. Additionally, the interface is currently tailored to the specific domain mentioned in the paper (Beauty dataset), and future work should aim to expand its applicability across several domains to ensure broader usability. While the web application is continuously improving, future efforts will focus on implementing new features, including sections for model uploads. For future work, we want to evaluate the web app interface developed in this study with human participants to ensure that the interface is intuitive and user-friendly, highlighting possible areas of improvement.

[6]https://huggingface.co/facebook/bart-large-mnli

## References


[1] G. Bonetta, C. D. Hromei, L. Siciliani, M. A. Stranisci, Preface to the Eighth Workshop on Natural Language for Artificial Intelligence (NL4AI), in: Proceedings of the Eighth Workshop on Natural Language for Artificial Intelligence (NL4AI 2024) co-located with 23th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2024), 2024.

[2] Z. Ping, G. Sang, Z. Liu, Y. Zhang, Aspect category sentiment analysis based on prompt-based learning with attention mechanism, Neurocomputing 565 (2024) 126994.

[3] W. Liao, B. Zeng, X. Yin, P. Wei, An improved aspect-category sentiment analysis model for text sentiment analysis based on roberta, Appl. Intell. 51 (2021) 3522–3533.

[4] H. Yang, K. Li, PyABSA, 2023. URL: https://github.com/yangheng95/PyABSA.

[5] J. Liu, Z. Teng, L. Cui, H. Liu, Y. Zhang, Solving aspect category sentiment analysis as a text generation task, in: EMNLP (1), Association for Computational Linguistics, 2021, pp. 4406–4416.

[6] W. Zhang, X. Li, Y. Deng, L. Bing, W. Lam, A survey on aspect-based sentiment analysis: Tasks, methods, and challenges, CoRR abs/2203.01054 (2022). doi:10.48550/arXiv.2203.01054.

[7] L. D. Quilio, F. Fioravanti, Evaluating the aspect-category-opinion-sentiment analysis task on a custom dataset (short paper), in: NL4AI@AI*IA, volume 3551 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.

[8] SemEval, Semeval-2016 task 5, 2016. URL: https://alt.qcri.org/semeval2016/task5/.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (2020) 140:1–140:67.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: NIPS, 2017, pp. 5998–6008.

[11] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Y. Zhao, Y. Huang, A. M. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, J. Wei, Scaling instruction-finetuned language models, CoRR abs/2210.11416 (2022).

[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: NeurIPS, 2019, pp. 8024–8035.