EFFICIENT ASPECT-OPINION CONNECTION WITH R-GAT FOR SENTIMENT ANALYSIS

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

by

20BCR7105 - VENNA JAYA DEVI SAI CHARAN

20BCI7116 - VISWANADHA LAKSHMI KALYANI

20BCE7591 - NIMMAGADDA VAMSI KRISHNA

20BCE7391 - AMRUTHA MACHARLA

Under the Guidance of

Dr. Hussain Syed

SCHOOL OF COMPUTER ENGINEERING

VIT-AP UNIVERSITY

AMARAVATI- 522237

December 2023

CERTIFICATE

This confirms that the Capstone Project, "EFFICIENT ASPECT-OPINION CONNECTION WITH R-GAT FOR SENTIMENT ANALYSIS," being turned in by **VENNA JAYA DEVI SAI CHARAN (20BCR7105)** fulfills the requirements for the honour of Bachelor of Technology. This is a documentation of genuine work completed under my supervision. This project's content, in its entirety or in portions, has not been lifted from any other source, nor has it been submitted to another university or institute in hopes of receiving a degree or other recognition; the equivalency has been verified.

[illegible]

Dr. HUSSAIN SYED

Guide

The thesis is [✓ satisfactory]/unsatisfactory

[illegible]

PROF. GOPIKRISHNAN S VINTA

Internal Examiner 1

[illegible]

PROF. SURENDRA REDDY

Internal Examiner 2

Approved by

[illegible]

Dr. G Muneeswari

HoD, Data Science and Engineering

School of Computer Science and Engineering

CERTIFICATE

This confirms that the Capstone Project, "EFFICIENT ASPECT-OPINION CONNECTION WITH R-GAT FOR SENTIMENT ANALYSIS," being turned in by **VISWANADHA LAKSHMI KALYANI (20BCI7116)** fulfills the requirements for the honour of Bachelor of Technology. This is a documentation of genuine work completed under my supervision. This project's content, in its entirety or in portions, has not been lifted from any other source, nor has it been submitted to another university or institute in hopes of receiving a degree or other recognition; the equivalency has been verified.

[illegible]

Dr. HUSSAIN SYED

Guide

The thesis is [✓ satisfactory]/unsatisfactory

[illegible]

PROF. GOPIKRISHNAN S VINTA

Internal Examiner 1

[illegible]

PROF. SURENDRA REDDY

Internal Examiner 2

Approved by

[illegible] 8/12/23

~~Dr. R S Reeja~~ Dr. Reeja. S.R

HoD, Artificial Intelligence & Machine Learning

School of Computer Science and Engineering

CERTIFICATE

This confirms that the Capstone Project, "EFFICIENT ASPECT-OPINION CONNECTION WITH R-GAT FOR SENTIMENT ANALYSIS," being turned in by **AMRUTHA MACHARLA (20BCE7391)** fulfills the requirements for the honour of Bachelor of Technology. This is a documentation of genuine work completed under my supervision. This project's content, in its entirety or in portions, has not been lifted from any other source, nor has it been submitted to another university or institute in hopes of receiving a degree or other recognition; the equivalency has been verified.

[illegible]

Dr. HUSSAIN SYED

Guide

The thesis is [✓ satisfactory]/unsatisfactory

[illegible]

PROF. GOPIKRISHNAN S VINTA

Internal Examiner 1

[illegible]

PROF. SURENDRA REDDY

Internal Examiner 2

Approved by

[illegible]

Dr. G Muneeswari

HoD, Data Science and Engineering

School of Computer Science and Engineering

CERTIFICATE

This confirms that the Capstone Project, "EFFICIENT ASPECT-OPINION CONNECTION WITH R-GAT FOR SENTIMENT ANALYSIS," being turned in by **NIMMAGADDA VAMSI KRISHNA (20BCE7591)** fulfills the requirements for the honour of Bachelor of Technology. This is a documentation of genuine work completed under my supervision. This project's content, in its entirety or in portions, has not been lifted from any other source, nor has it been submitted to another university or institute in hopes of receiving a degree or other recognition; the equivalency has been verified.

[illegible]

Dr. HUSSAIN SYED

Guide

The thesis is [✓ satisfactory]/unsatisfactory

[illegible] 8/12/23

PROF. GOPIKRISHNAN S VINTA

Internal Examiner 1

[illegible]

PROF. SURENDRA REDDY

Internal Examiner 2

Approved by

[illegible]

Dr. G Muneeswari

HoD, Data Science and Engineering

School of Computer Science and Engineering

ACKNOWLEDGEMENTS

We express our gratitude to Dr Hussain Syed, our mentor, for his unwavering encouragement and assistance during the whole endeavour. He would frequently ask me questions about specific parts of the project that would help me figure out why and how to solve particular problems. His unwavering support has had a great effect on me, and his concepts are reflected in the things I do. We also want to thank the School of Computer Science and Engineering for giving me the chance to improve my abilities and expand my skill set, both of which will undoubtedly help me in the years to come.

A project serves as a scaffold between theoretical and practical learning. With this in mind, we worked hard and made progress on the endeavour, with the support and encouragement of everyone in our immediate vicinity.

Yes, I would also want to thank my supervisors and friends for their encouragement and support in helping me organize and come up with creative solutions for my assignments. I am really appreciative of each of these. We were able to complete this work and make it a wonderful and enjoyable experience because of them. This project is the result of many people's tireless labour, without which it would not have been feasible.

ABSTRACT

The sentiment polarity on several elements of online reviews is examined in Efficient Aspect-Opinion Connection with R-GAT for Sentiment Analysis. Most recent approaches utilize attention-based neural network models to implicitly connect features with opinion words. Nevertheless, due to the intricacy of the language and the existence of several elements in a single phrase, these models often confuse the links. In this paper, we address this problem using effective syntactic information encoding. Bending and pruning a standard dependency parse tree is the first step towards building a unified aspect-oriented dependency tree structure.

An anchor point for this structure is a target aspect. We then propose to encode the new tree structure for sentiment prediction using a relational graph attention network (R-GAT). The results of the experiment, the IMBD, Yelp, and Amazon cells data sets all corroborate the notion that our method may be used to more successfully create linkages between attributes and opinion words. Consequently, there is a noticeable improvement in the graph attention network's (GAT) performance.

TABLE OF CONTENTS

List of Figures

CHAPTER 1 INTRODUCTION

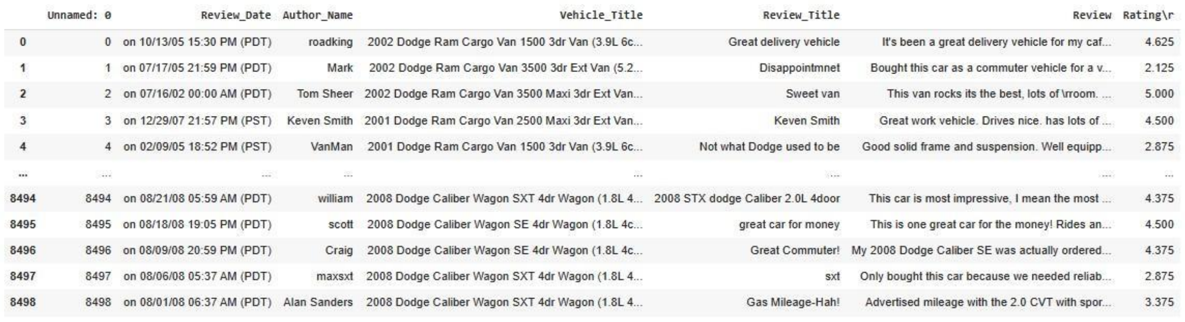
There are more strong opinions than ever in our world. With any service we utilize, we are either happy or unhappy. Thanks to social media, we can now instantly express our ideas to the public. There are several forms of data sources accessible, like as reviews, customer satisfaction questionnaires, customer complaints, etc. By using this data, companies may decide how to enhance their offers and have a deeper understanding of what their customers are saying.

Sentiment analysis is a procedure used to determine how satisfied a consumer is with a service. The degree of satisfaction is usually quantified into three categories: positive, negative, and neutral. Despite the possibility of varying class sizes. Labelled data may be utilized with a variety of machine and deep-learning classification methods.

We would have to go back to methods based on lexicons. Though this essay will only cover supervised methods, there are many more ways to do sentiment analysis unsupervised.

This post's USP is Aspect Based Sentiment Analysis (ABSA). Since consumers discuss several parts of the service in real-world data and their sentiments might vary depending on whatever component they are discussing, this is the most practical application of sentiment analysis.

In this restaurant review, the customer talks about three things: the pricing, the cuisine, and the wait. It is consequently nonsensical to have a single feeling running across the entire review. Aspect-based sentiment analysis allows us to learn more about customer satisfaction and provide useful recommendations for the restaurant, like adding extra staff to reduce wait times.



	Unnamed: 0	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating\r
0	0	on 10/13/05 15:30 PM (PDT)	roadking	2002 Dodge Ram Cargo Van 1500 3dr Van (3.9L 6c...	Great delivery vehicle	It's been a great delivery vehicle for my caf...	4.625
1	1	on 07/17/05 21:59 PM (PDT)	Mark	2002 Dodge Ram Cargo Van 3500 3dr Ext Van (5.2...	Disappointmnet	Bought this car as a commuter vehicle for a v...	2.125
2	2	on 07/16/02 00:00 AM (PDT)	Tom Sheer	2002 Dodge Ram Cargo Van 3500 Maxi 3dr Ext Van...	Sweet van	This van rocks its the best, lots of \rroom. ...	5.000
3	3	on 12/29/07 21:57 PM (PST)	Keven Smith	2001 Dodge Ram Cargo Van 2500 Maxi 3dr Ext Van...	Keven Smith	Great work vehicle. Drives nice. has lots of ...	4.500
4	4	on 02/09/05 18:52 PM (PST)	VanMan	2001 Dodge Ram Cargo Van 1500 3dr Van (3.9L 6c...	Not what Dodge used to be	Good solid frame and suspension. Well equipp...	2.875
...	...	...	...	...	...	...	...
8494	8494	on 08/21/08 05:59 AM (PDT)	william	2008 Dodge Caliber Wagon SXT 4dr Wagon (1.8L 4...	2008 STX dodge Caliber 2.0L 4door	This car is most impressive, I mean the most ...	4.375
8495	8495	on 08/18/08 19:05 PM (PDT)	scott	2008 Dodge Caliber Wagon SE 4dr Wagon (1.8L 4c...	great car for money	This is one great car for the money! Rides an...	4.500
8496	8496	on 08/09/08 20:59 PM (PDT)	Craig	2008 Dodge Caliber Wagon SE 4dr Wagon (1.8L 4c...	Great Commuter!	My 2008 Dodge Caliber SE was actually ordered...	4.375
8497	8497	on 08/06/08 05:37 AM (PDT)	maxsxt	2008 Dodge Caliber Wagon SXT 4dr Wagon (1.8L 4...	sxt	Only bought this car because we needed reliab...	2.875
8498	8498	on 08/01/08 06:37 AM (PDT)	Alan Sanders	2008 Dodge Caliber Wagon SXT 4dr Wagon (1.8L 4...	Gas Mileage-Hah!	Advertised mileage with the 2.0 CVT with spor...	3.375

Data set representation.

1.1 OBJECTIVES

- In this paper, we address this problem using effective syntactic information encoding. Bending and pruning a standard dependency parse tree is the first step towards building a unified aspect-oriented dependency tree structure.
- An anchor point for this structure is a target aspect. We then propose to encode the new tree structure for sentiment prediction using a relational graph attention network (R- GAT).
- The results of the experiments, IMBD, Yelp, and Amazon cells data sets all corroborate the notion that our method may be used to build
- relationships between features and viewpoint terms more successfully. Consequently, there is a noticeable improvement in the graph attention network's (GAT) performance.

1.2 Background and Literature Survey

A group of texts (such as product reviews or comments from social media) describing a certain object (such as a new mobile phone model) is fed into Aspect Based Sentiment Analysis (ABSA) systems. The systems try to identify the primary (i.e., most spoken about) characteristics of the entity (i.e., "battery," "screen") and to calculate the average sentiment of the texts for each characteristic (i.e., the average degree of positivity or negativity for each aspect). Despite the fact that a number of ABSA systems—mostly research prototypes—have been developed, there is neither a task decomposition for ABSA nor a set of defined assessment metrics for the subtasks that ABSA systems must complete (Liu, 2012).

The three primary subtasks in this thesis' novel task decomposition for ABSA are aspect term extraction, aspect term aggregation, and aspect term polarity estimation. The first subtask identifies single- and multi-word phrases (such as "hard disc" and "battery") that name elements of the thing under discussion; these terms are referred to as aspect terms from here on. In the second subtask, comparable aspect words (like "price" and "cost") are aggregated (clusters) based on user choices and other constraints (like the screen size where the ABSA system results would be displayed). The average sentiment for each aspect phrase or aspect term cluster is estimated in the third subtask.

During the course of this thesis, benchmark data sets for various types of entities (such as laptops and restaurants) were created for each of the aforementioned subtasks. For every subtask, new assessment metrics are presented with the justification that they are superior to earlier metrics. The thesis also suggests new approaches (or enhancements over existing ones) for each subtask, demonstrating empirically on the built benchmark data sets that the new approaches (or the upgraded versions) are superior to state-of-the-art approaches, or at least equivalent to them. The author first contributed to the creation of a system that assesses the sentiment (positive, negative, or neutral) of whole communications (such as tweets) in order to determine aspect word polarity. Two classifiers form the basis of the system (SVM), one that recognizes sentiment in messages (good or negative), and another that separates the former from the latter. The system can handle uneven sentiment classes (neutral messages can frequently be as many as the positive and negative ones combined) because of the usage of two classifiers. The sentiment scores that are automatically appended to the lexical items and the sentiment lexical's current characteristics are used by the classifiers. With excellent results, the sentiment estimate system took part in international competitions (Sentiment Analysis on the Twitter job). The system's excellent generalization ability was further demonstrated by the findings, which also revealed that it outperformed the majority of the other rivals when applied to communications that differed from those observed during training (such as SMS messages rather than tweets). In order to estimate the sentiment for each aspect word rather than for each whole message (such as a sentence), the message-level sentiment estimation method from the previous paragraph was then evaluated using aspect term polarity estimation data sets that were created throughout the work for this thesis.

In this instance, the system from the previous paragraph was applied to categorize each complete message into a sentiment class (positive, negative, or neutral). Assuming that every aspect term in a message carries the same sentiment, the sentiment class of the message was also assigned to every aspect term. While some messages do have aspect words with various polarity (one positive and one negative aspect term, for example), an investigation revealed that these types of communications are very uncommon, at least in the experiment data sets. Consequently, despite not having to be retrained (unlike the competitors), the system works admirably when compared to other systems. For the subtask of aspect term polarity estimation, a new evaluation measure was also proposed, which considers the following: (i) misclassifying a positive aspect term into the negative class, for example, is a bigger mistake than misclassifying it into the neutral category; and (ii) since frequent aspect terms are of primary interest to end users of ABSA systems, errors that do not significantly affect the average polarity scores of frequent aspect terms are not really significant. The message-level system in the preceding paragraph performs even better than its rivals using the new assessment metric.

1.3 Organization of the Report

The project report's following chapters are explained as follows:

The suggested structure, methodology, tools, and programming details are included in Section 2.

The costs related to carrying out the project are listed in Section 3.

Section 4 looks at the results obtained once the project was completed.

The report is concluded in Chapter 5.

Codes are covered in Chapter 6.

References are included in Chapter 7.

CHAPTER 2

Working model and How it is implemented using various data sets.

2.1.Cleaning the text

Preprocessing the text is the most crucial stage in solving any NLP issue. This may consist of:

lower case

Take out any non-ASCII characters.

Take out the punctuation

Eliminate superfluous spaces and lines.

Eliminate stop words.

Rooting

simple spelling correction

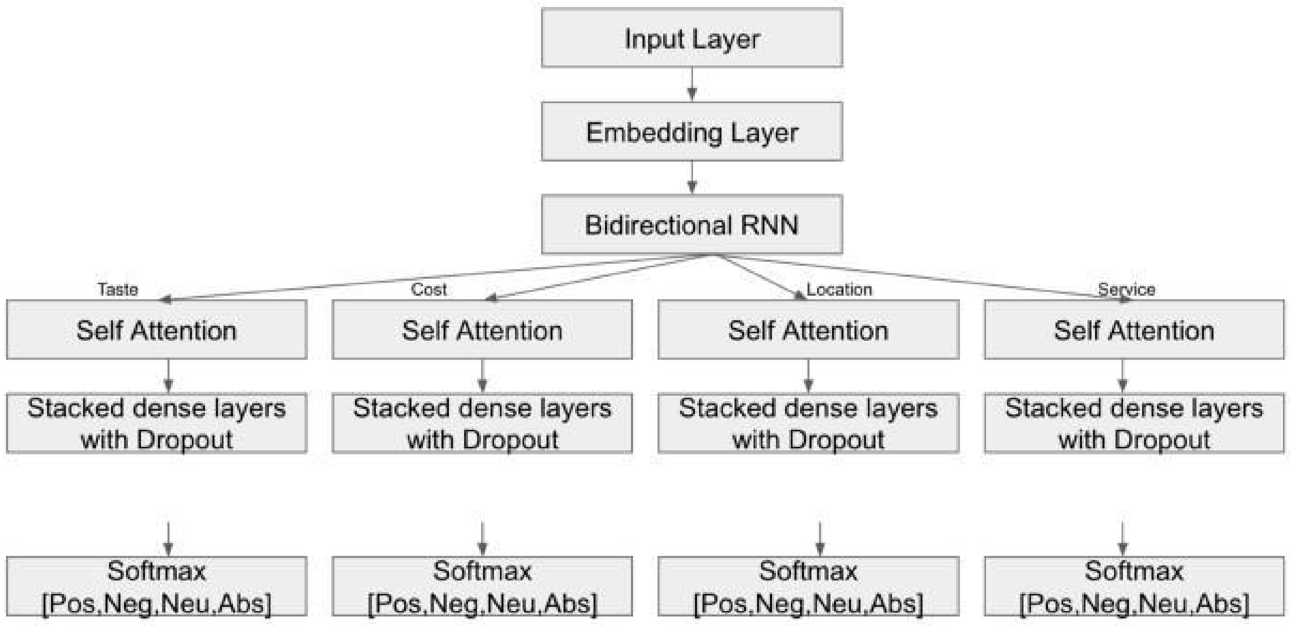
Correcting slang, etc.

For most processes, Python programs like nltk and space may be utilized. Using a dictionary of spelling mistakes and their corresponding repairs is a more straightforward method for spell correction. The handling of emoticons is another part of preprocessing. You may not need to use emoticons for any other NLP activity. They are essential for sentiment analysis, though. To identify, eliminate, or swap out the emoticon for a reserved token, use the Python remote module.

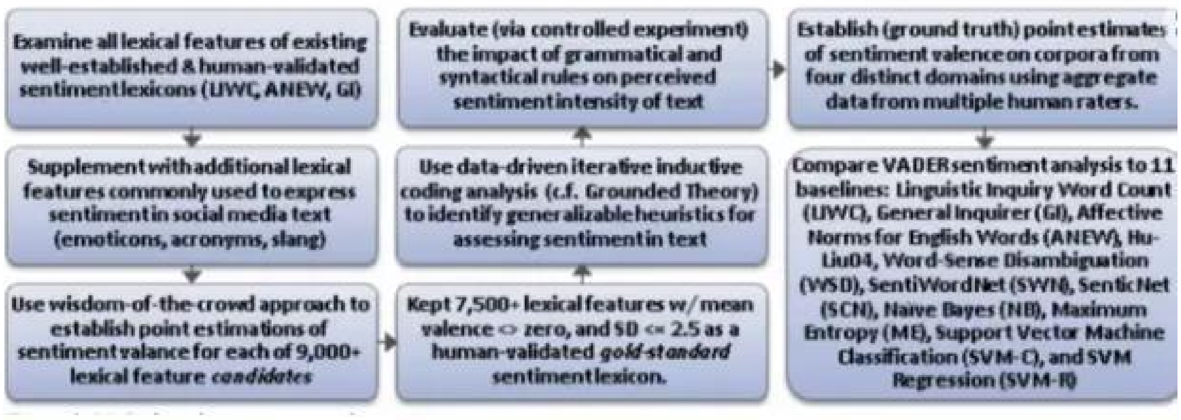
2.2 Model

We have reformulated our challenge as a multi-task classification problem based on the aforementioned data. Every element alludes to a distinct activity, which is categorized into four classes. The classes are: Positive, Negative, Neutral, and Absent, where the latter denotes the absence of the feature from the evaluation.

This is how the model architecture appears:


Input Layer
Embedding Layer
Bidirectional RNN
Taste
Cost
Location
Service
Self Attention
Self Attention
Self Attention
Self Attention
Stacked dense layers with Dropout
Stacked dense layers with Dropout
Stacked dense layers with Dropout
Stacked dense layers with Dropout
Softmax [Pos,Neg,Neu,Abs]
Softmax [Pos,Neg,Neu,Abs]
Softmax [Pos,Neg,Neu,Abs]
Softmax [Pos,Neg,Neu,Abs]

An Aspect Based Sentiment Analysis Model Architecture


Examine all lexical features of existing well-established & human-validated sentiment lexicons (LIWC, ANEW, GI)
Supplement with additional lexical features commonly used to express sentiment in social media text (emoticons, acronyms, slang)
Use wisdom-of-the-crowd approach to establish point estimations of sentiment valance for each of 9,000+ lexical feature candidates
Kept 7,500+ lexical features w/ mean valence <> zero, and SD <= 2.5 as a human-validated gold-standard sentiment lexicon.
Use data-driven iterative inductive coding analysis (c.f. Grounded Theory) to identify generalizable heuristics for assessing sentiment in text
Evaluate (via controlled experiment) the impact of grammatical and syntactical rules on perceived sentiment intensity of text
Establish (ground truth) point estimates of sentiment valence on corpora from four distinct domains using aggregate data from multiple human raters.
Compare VADER sentiment analysis to 11 baselines: Linguistic Inquiry Word Count (LIWC), General Inquirer (GI), Affective Norms for English Words (ANEW), Hu-Liu04, Word-Sense Disambiguation (WSD), SentiWordNet (SWN), SenticNet (SCN), Naive Bayes (NB), Maximum Entropy (ME), Support Vector Machine Classification (SVM-C), and SVM Regression (SVM-R)

I run the tokenized input via an embedding layer and a bidirectional LSTM in order to build the model. I then create four branches in my model, one for each facet, after that. Every branch contains a self-attention layer that increases the weight of words that describe the specified feature. It's also possible to test without the self-attention layer. However, my research revealed that the model was able to learn the patterns and n-grams written for each component by applying attention in a localized way for each aspect. I've applied a straightforward dot product focus here. This may be altered to use a variety of complex attention techniques (which I will discuss in my next postings!).

- Next, we apply two or three totally linked thick layers. A soft-max is used to change each branch's output into one of four classes: positive, negative, neutral, or nonexistent. I believe this architecture to be straightforward but efficient, even though there are undoubtedly many more models for aspect-based sentiment analysis. It not only does away with the requirement for separate models for each aspect, but it also trains the aspect's sentiment in addition to whether or not it exists.
- This study makes the following contributions: By reshaping and pruning standard dependency trees to concentrate on the desired aspects, we offer an aspect-oriented tree structure.
- In order to encode the dependence linkages and determine the relationships between aspects and opinion words, we suggest a novel GAT model.
- This work's source code is available for use in other investigations.

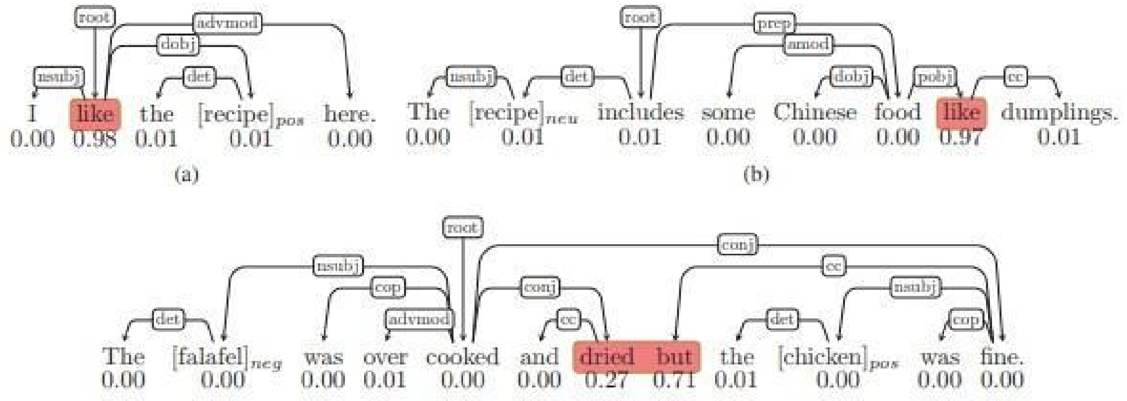


Fig: Three examples from restaurant reviews to illustrate the relationships among aspect, attention, and syntax in ABSA.

2.3. Aspect-Oriented Dependency Tree

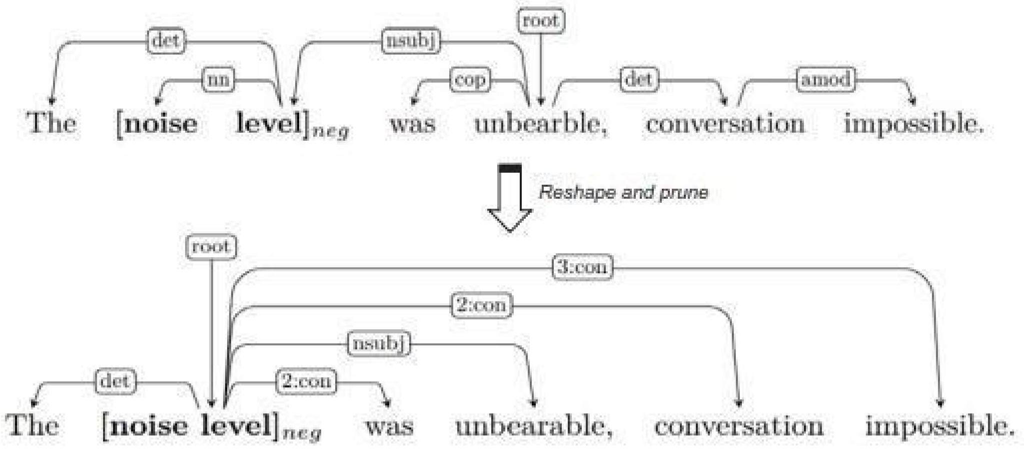
We go into further depth about how to build an aspect-oriented dependency tree in this section.

2.3.1 Aspect, Attention and Syntax

Dependency parsing, which creates a dependency tree to reflect the grammatical structure, can reveal the syntactic structure of a phrase. Labels and directed edges can be used to indicate the connections between words. As seen in Figure 1, we utilize three examples to demonstrate how aspect, attention, and syntax relate to one another in ABSA. The attention-based LSTM model correctly attends to the aspect recipe in the first example, where the verb "like" is employed to convey a positive mood. Nevertheless, the model continues to give it a high weight when it is used as a preposition in the second case, which leads to an incorrect prediction. The third example illustrates a situation in which a single statement has two features with opposing feeling polarity. The LSTM model incorrectly gives the terms but and dried high attention weights for the aspect chicken, which results in yet another incorrect prediction. By adding clear syntactic links between aspects and other words, such errors are likely to be prevented. In the third scenario, for instance, things may have turned out differently if the model had identified the direct dependence link between chicken and fine instead of with but.

3.2 Aspect-Oriented Dependency Tree

According to the aforementioned study, dependency relations that are directly associated with an aspect should be given more weight than other relations since they may help a model concentrate more on opinion terms that are relevant to it. Furthermore, as seen in Figure 1, a dependency tree is typically not anchored at a target aspect and contains a wealth of grammatical information. However, ABSA focuses on a specific feature rather than the tree's base. Motivated by the aforementioned discoveries, we reshape an original dependency tree to root it at a desired aspect, then prune the tree to remove redundant relations. This results in a unique aspect-oriented dependency tree structure. The aforesaid procedure is described by Algorithm 1. In order to retrieve the dependency tree for an input phrase, we first apply a dependency parser, where rij represents the dependency connection from node I to j. Next, we construct a three-step aspect-oriented dependency tree. First, multiple-word aspects are considered entities, and the target aspect is placed at the root. Second, we designate the original nodes as the children, for which they have direct connections to the aspect.


det
nn
nsubj
root
cop
det
amod
The [noise level]neg was unbearble, conversation impossible.
Reshape and prune
root
3:con
2:con
nsubj
det
2:con
The [noise level]neg was unbearable, conversation impossible.

Building an aspect-oriented dependency tree from a standard dependency tree (top) (bottom).

4. Model Training

The word embedding of tree nodes is encoded using Bi LSTM, and its output hidden state hi is obtained for the initial representation h0 i of leaf node i. After that, the aspect words are encoded using a second Bi LSTM, and the initial representation h0 of this root is derived from its average hidden state. Following the application of R-GAT to an aspect-oriented tree, the root representation hl of the tree is mapped to probabilities across the various emotion polarities after passing via a fully connected soft-max layer.

Soft-max(Wphl a + bp) = p(a) .. (i)

Finally, our goal function is the conventional cross-entropy loss:

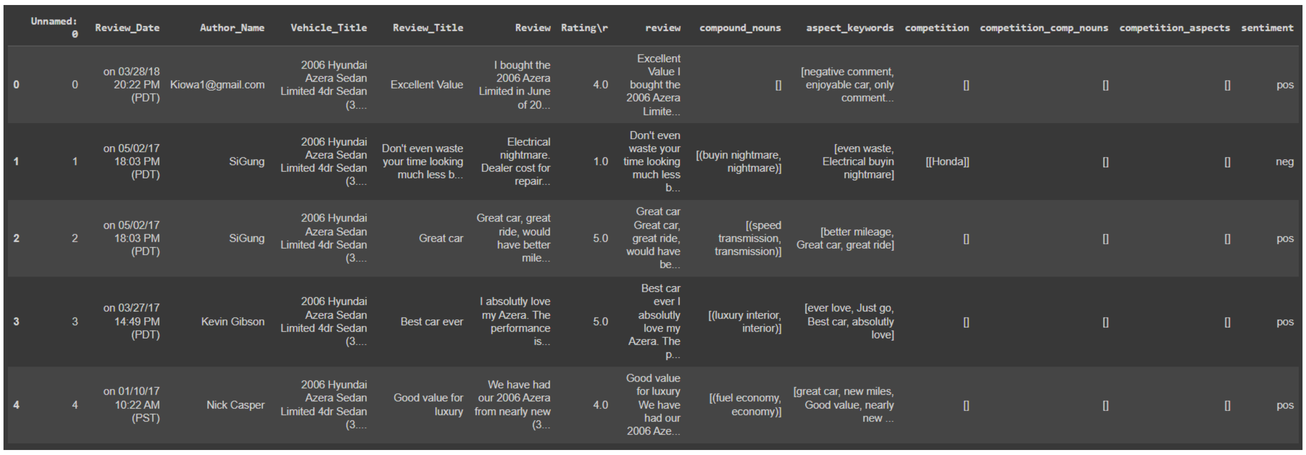
$L(\theta) = -$ E(S, A) ED E an E A log p(a).....(ii) in where D is the set of all sentence-aspect pairings, A is the set of aspects that appear in sentence S, and θ is the set of all trainable parameters.

CHAPTER 3 COST ANALYSIS

Our model does not incorporate any costs; we have just used publicly available data sets.

CHAPTER 4

RESULTS AND DISCUSSIONS



	Unnamed: 0	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating\r	review	compound_nouns	aspect_keywords	competition	competition_comp_nouns	competition_aspects	sentiment
0	0	on 03/28/18 20:22 PM (PDT)	Kiowa1@gmail.com	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Excellent Value	I bought the 2006 Azera Limited in June of 20...	4.0	Excellent Value I bought the 2006 Azera Limite...	[]	[negative comment, enjoyable car, only comment...	[]	[]	[]	pos
1	1	on 05/02/17 18:03 PM (PDT)	SiGung	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Don't even waste your time looking much less b...	Electrical nightmare. Dealer cost for repair...	1.0	Don't even waste your time looking much less b...	[(buyin nightmare, nightmare)]	[even waste, Electrical buyin nightmare]	[[Honda]]	[]	[]	neg
2	2	on 05/02/17 18:03 PM (PDT)	SiGung	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Great car	Great car, great ride, would have better mile...	5.0	Great car Great car, great ride, would have be...	[(speed transmission, transmission)]	[better mileage, Great car, great ride]	[]	[]	[]	pos
3	3	on 03/27/17 14:49 PM (PDT)	Kevin Gibson	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Best car ever	I absolutly love my Azera. The performance is...	5.0	Best car ever I absolutly love my Azera. The p...	[(luxury interior, interior)]	[ever love, Just go, Best car, absolutly love]	[]	[]	[]	pos
4	4	on 01/10/17 10:22 AM (PST)	Nick Casper	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Good value for luxury	We have had our 2006 Azera from nearly new (3...	4.0	Good value for luxury We have had our 2006 Aze...	[(fuel economy, economy)]	[great car, new miles, Good value, nearly new ...	[]	[]	[]	pos

Fig: Data set and their reviews

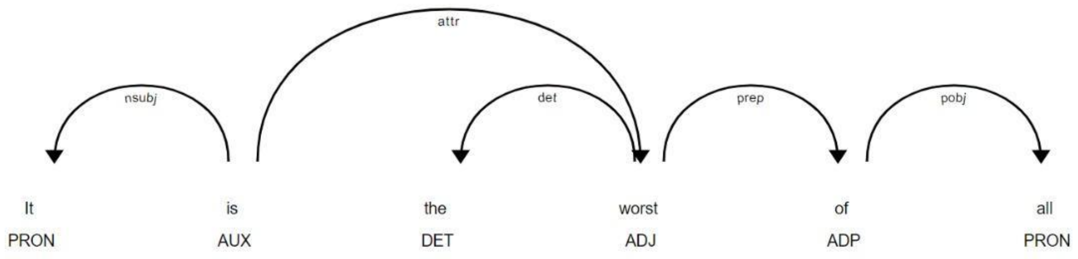


Figure: Pronoun, Adjective, Auxiliary, Determiner, and Ad Position Differentiation

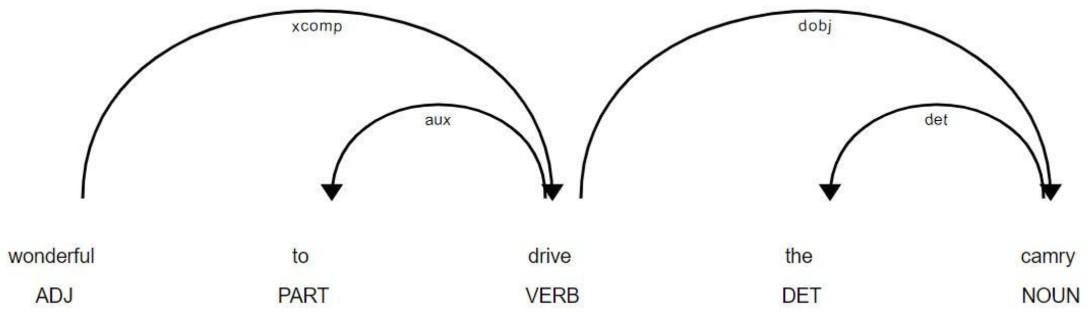


Fig: Comparison between the compound words

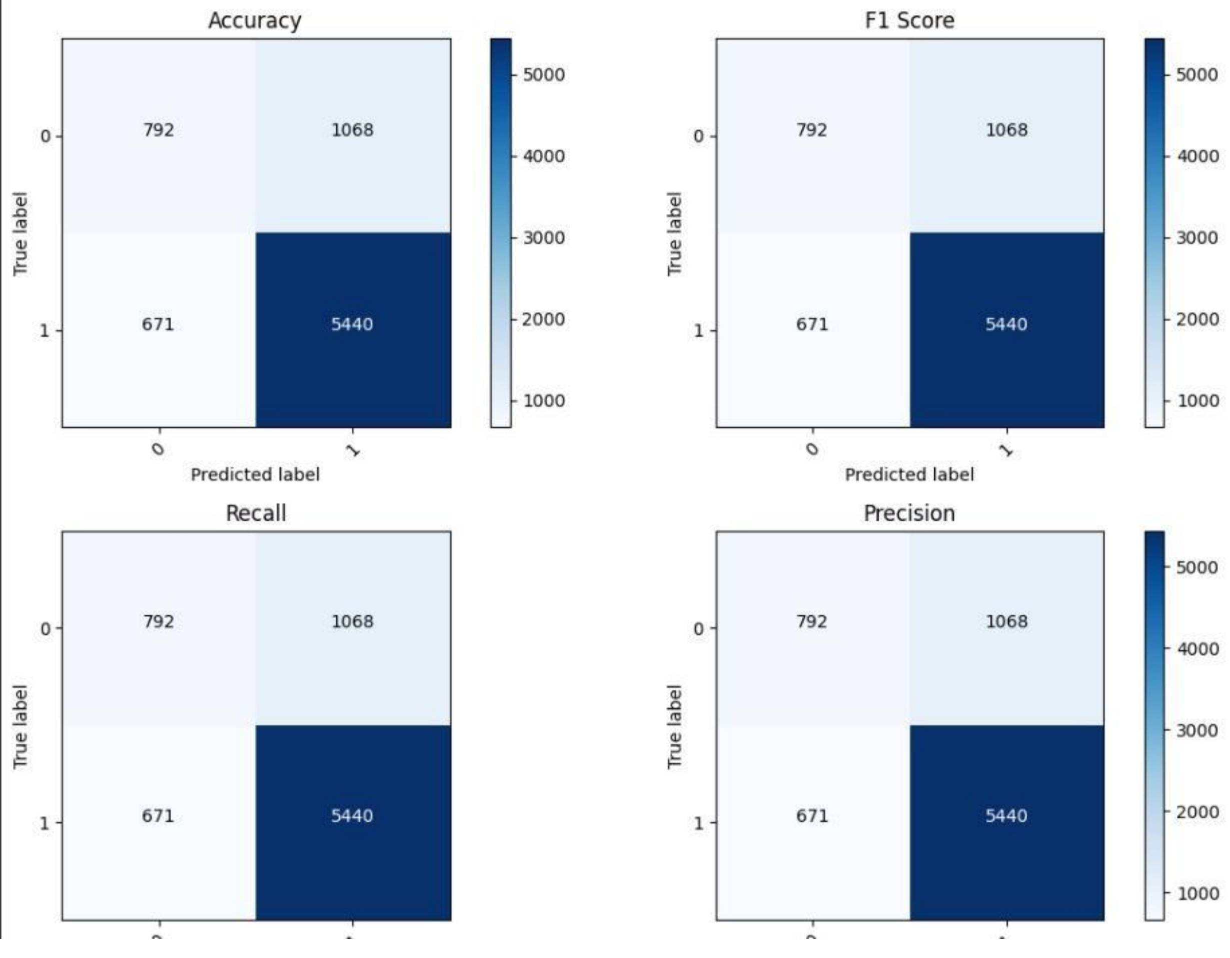
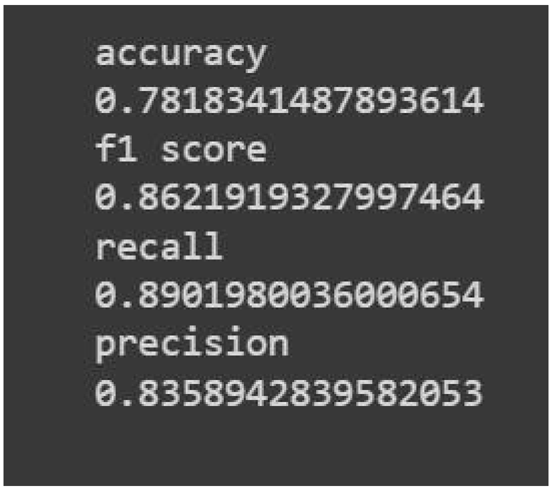
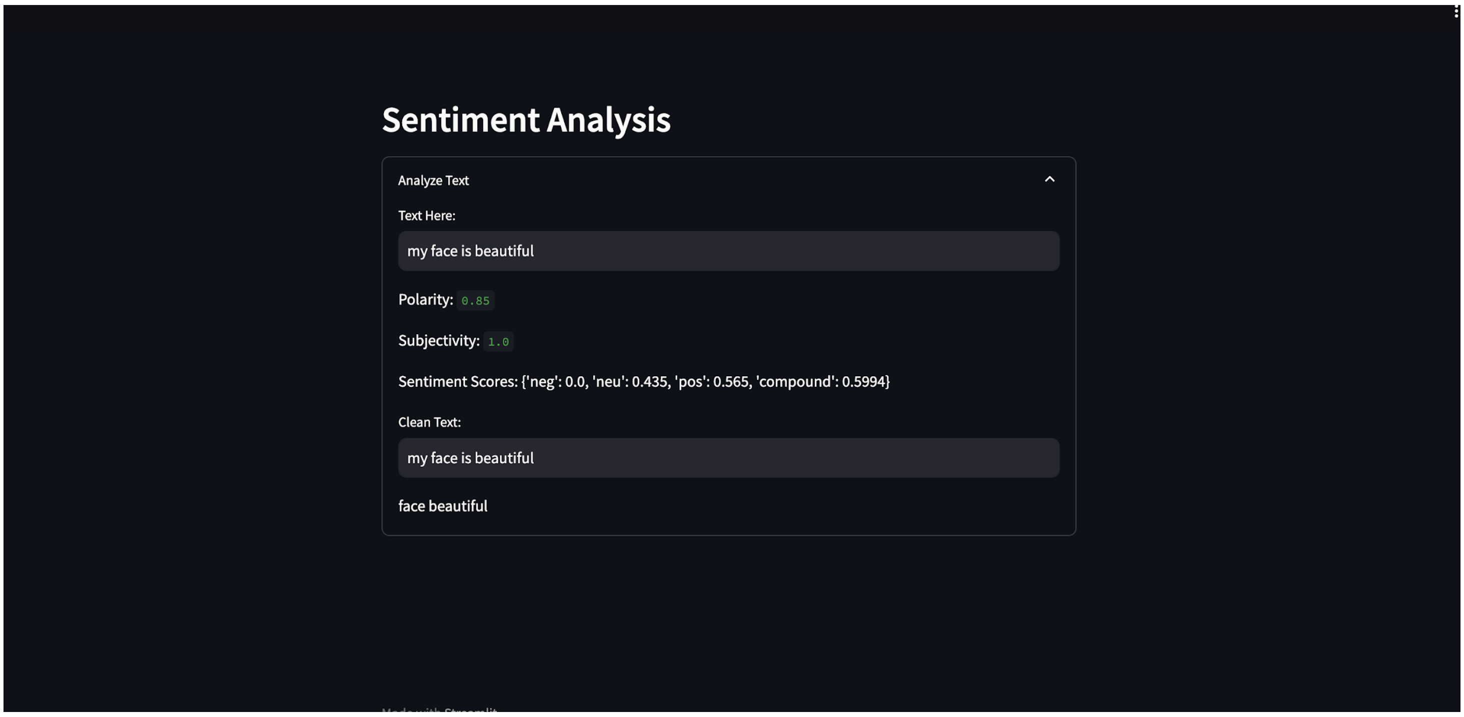


Fig: Graphs of Accuracy, Score, Recall, Precision.



```
accuracy
0.7818341487893614
f1 score
0.8621919327997464
recall
0.8901980036000654
precision
0.8358942839582053
```

Fig: Accuracy, Score, Recall, Precision.



Sentiment Analysis

Analyze Text

Text Here:

my face is beautiful

Polarity: `0.85`

Subjectivity: `1.0`

Sentiment Scores: {'neg': 0.0, 'neu': 0.435, 'pos': 0.565, 'compound': 0.5994}

Clean Text:

my face is beautiful

face beautiful

Fig: Result displayed in web application

CHAPTER 5

CONCLUSION AND FUTURE WORK

types of entities were created in the process of writing this thesis. For every subtask, new assessment metrics were added with the justification that they were more suited than earlier metrics. The thesis offered new techniques (or improvements over existing methods) for each subtask, and it was demonstrated empirically on the built benchmark data sets that the new methods (or the upgraded versions) are superior to state-of-the-art approaches, or at least similar to them. Additionally, a new assessment metric for the subtask of aspect term polarity estimation was suggested, considering that (i) incorrectly categorizing a positive aspect phrase into the negative class, for instance, is a more serious error than incorrectly classifying it into the neutral category; and (ii) that errors that have no effect on the average polarity scores of common aspect phrases are not very significant to the end users of ABSA systems. The message-level system in the preceding paragraph performs even better than its rivals using the new assessment metric. When the new measure is applied instead of the competition's official measurements, the ranking of the best systems that took part in the appropriate subtask of an international competition is likewise altered. The data sets developed for this subtask throughout the writing of this thesis were also utilized in the competition, but somewhat modified. To enable the drawing of safer conclusions, future research in the field of ABSA should build even more benchmark data sets from a wider range of fields. A last step in the ABSA decomposition of this thesis that summarizes each aspect term (or collection of aspect terms) would be interesting to include as well. More precisely, comparing (only) unsupervised systems that extract aspect terms would be useful in ATE. Cluster labelling would be a very useful addition to aspect aggregation. An automated label extraction process might be used to represent the notion of all or most of the aspect words in each group (cluster) of aspect terms. Regarding sentiment estimate for aspect terms, It will be interesting to construct a regression system that, given a fresh batch of phrases (including an aspect term), returns a sentiment score rather than categorizing sentences having an aspect term and averaging the sentiment.

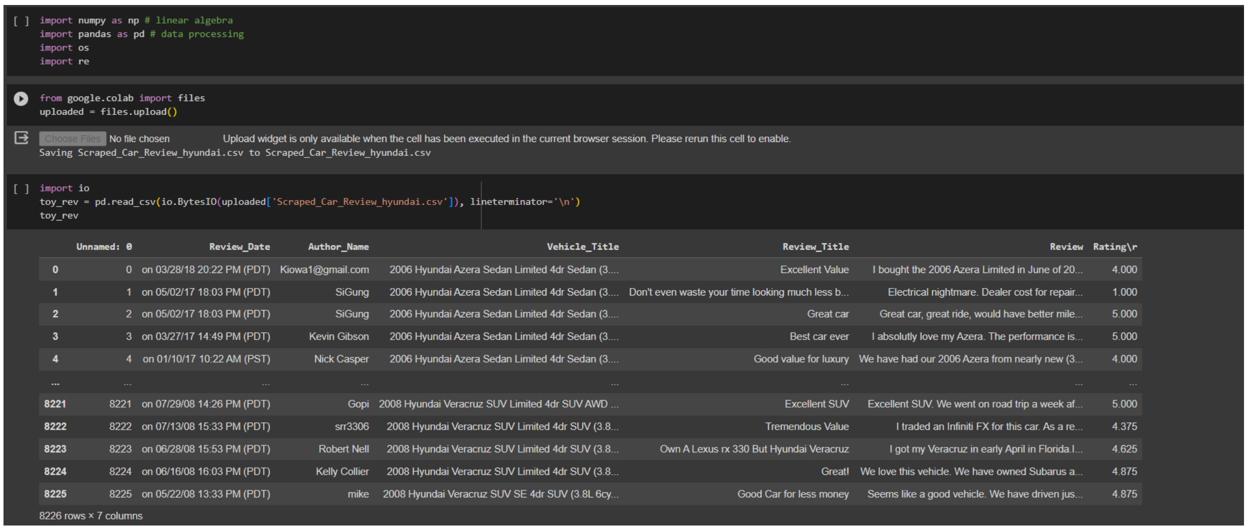
It will be interesting to construct a regression system that, given a fresh batch of phrases (including an aspect term), returns a sentiment score rather than categorizing sentences having an aspect term and averaging the sentiment.

CHAPTER 6

APPENDIX

For future reference, the entire source code is available at the following link:
https://colab.research.google.com/drive/1O_6g3yn_3n8iHKTUQeNMs-DR156f_FgI?usp=sharing#scrollTo=BMwXRTMr3KkP

Aspect-based sentiment analysis code.



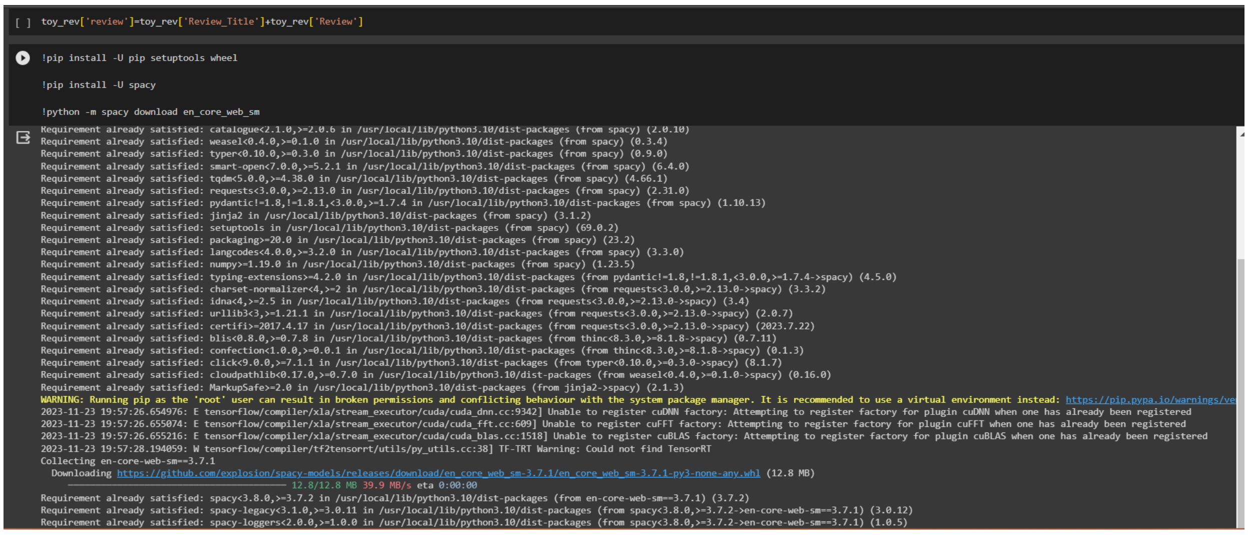
```
[ ] import numpy as np # linear algebra
    import pandas as pd # data processing
    import os
    import re
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Scraped_Car_Review_hyundai.csv to Scraped_Car_Review_hyundai.csv
```

```
[ ] import io
    toy_rev = pd.read_csv(io.BytesIO(uploaded['Scraped_Car_Review_hyundai.csv']), lineterminator='\n')
    toy_rev
```

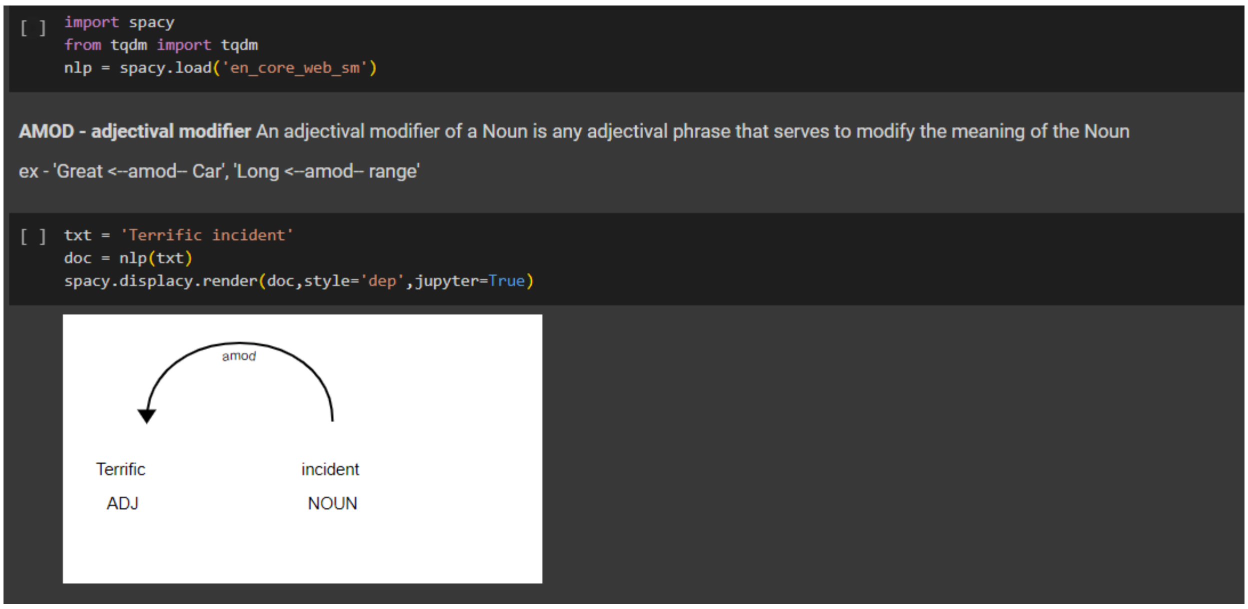
```
      Unnamed: 0              Review_Date       Author_Name                                   Vehicle_Title                            Review_Title                                           Review  Rating\r
0              0  on 03/28/18 20:22 PM (PDT)  Kiowa1@gmail.com   2006 Hyundai Azera Sedan Limited 4dr Sedan (3....                          Excellent Value   I bought the 2006 Azera Limited in June of 20...     4.000
1              1  on 05/02/17 18:03 PM (PDT)            SiGung   2006 Hyundai Azera Sedan Limited 4dr Sedan (3....  Don't even waste your time looking much less b...   Electrical nightmare. Dealer cost for repair...     1.000
2              2  on 05/02/17 18:03 PM (PDT)            SiGung   2006 Hyundai Azera Sedan Limited 4dr Sedan (3....                                Great car   Great car, great ride, would have better mile...     5.000
3              3  on 03/27/17 14:49 PM (PDT)      Kevin Gibson   2006 Hyundai Azera Sedan Limited 4dr Sedan (3....                            Best car ever   I absolutly love my Azera. The performance is...     5.000
4              4  on 01/10/17 10:22 AM (PST)       Nick Casper   2006 Hyundai Azera Sedan Limited 4dr Sedan (3....                    Good value for luxury  We have had our 2006 Azera from nearly new (3...     4.000
...          ...                         ...               ...                                             ...                                      ...                                              ...       ...
8221        8221  on 07/29/08 14:26 PM (PDT)              Gopi  2008 Hyundai Veracruz SUV Limited 4dr SUV AWD ...                            Excellent SUV   Excellent SUV. We went on road trip a week af...     5.000
8222        8222  on 07/13/08 15:33 PM (PDT)           srr3306   2008 Hyundai Veracruz SUV Limited 4dr SUV (3.8...                        Tremendous Value   I traded an Infiniti FX for this car. As a re...     4.375
8223        8223  on 06/28/08 15:53 PM (PDT)       Robert Nell   2008 Hyundai Veracruz SUV Limited 4dr SUV (3.8...      Own A Lexus rx 330 But Hyundai Veracruz    I got my Veracruz in early April in Florida.I...     4.625
8224        8224  on 06/16/08 16:03 PM (PDT)     Kelly Collier   2008 Hyundai Veracruz SUV Limited 4dr SUV (3.8...                                   Great!  We love this vehicle. We have owned Subarus a...     4.875
8225        8225  on 05/22/08 13:33 PM (PDT)              mike   2008 Hyundai Veracruz SUV SE 4dr SUV (3.8L 6cy...                    Good Car for less money   Seems like a good vehicle. We have driven jus...     4.875
8226 rows × 7 columns
```



```
[ ] toy_rev['review']=toy_rev['Review_Title']+toy_rev['Review']
```

```
!pip install -U pip setuptools wheel

!pip install -U spacy

!python -m spacy download en_core_web_sm
```

```
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.9.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.10.13)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy) (69.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.23.5)
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2023.7.22)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.3.0,>=8.1.8->spacy) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.3.0,>=8.1.8->spacy) (0.1.3)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.10.0,>=0.3.0->spacy) (8.1.7)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from weasel<0.4.0,>=0.1.0->spacy) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy) (2.1.3)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/ve
2023-11-23 19:57:26.654976: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2023-11-23 19:57:26.655074: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2023-11-23 19:57:26.655216: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2023-11-23 19:57:28.194059: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-none-any.whl (12.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 39.9 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in /usr/local/lib/python3.10/dist-packages (from en-core-web-sm==3.7.1) (3.7.2)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.0.5)
```



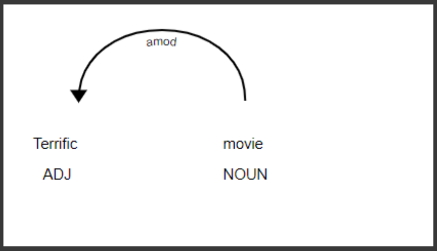
```
[ ] import spacy
    from tqdm import tqdm
    nlp = spacy.load('en_core_web_sm')
```

AMOD - adjectival modifier An adjectival modifier of a Noun is any adjectival phrase that serves to modify the meaning of the Noun

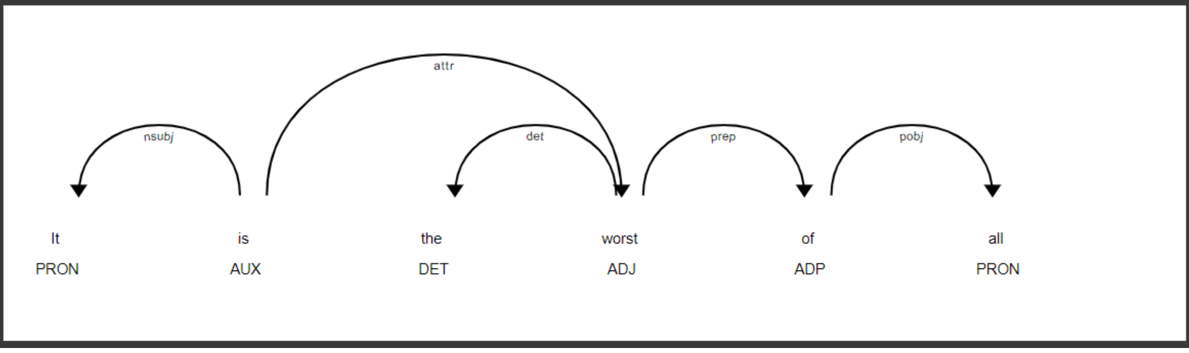
ex - 'Great <--amod-- Car', 'Long <--amod-- range'

```
[ ] txt = 'Terrific incident'
    doc = nlp(txt)
    spacy.displacy.render(doc,style='dep',jupyter=True)
```

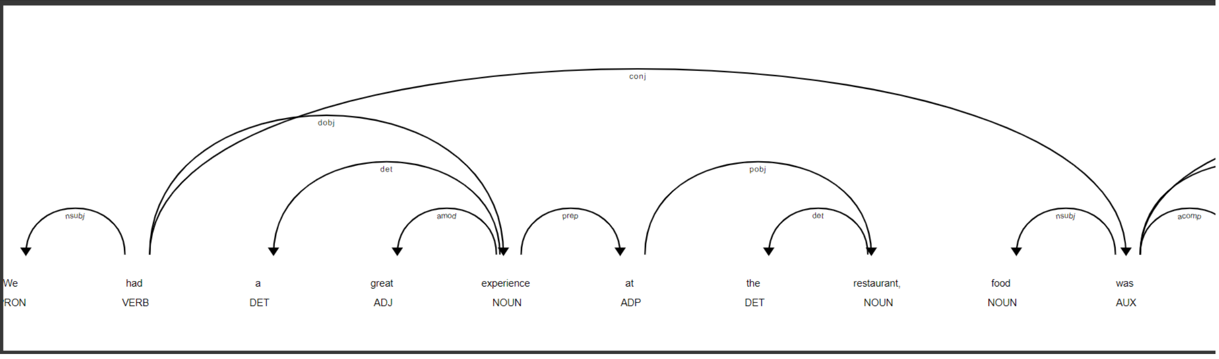
```
txt = 'Terrific movie'
doc = nlp(txt)
spacy.displacy.render(doc,style='dep',jupyter=True)
```



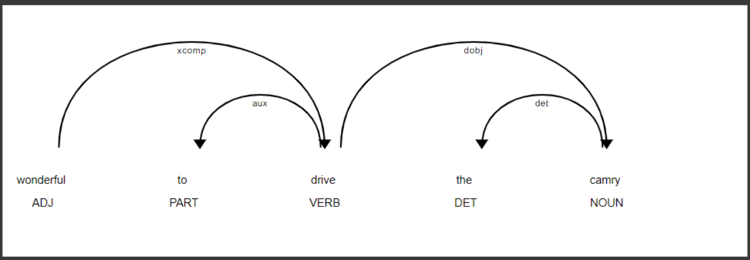
```
txt = 'It is the worst of all'
doc = nlp(txt)
spacy.displacy.render(doc,style='dep',jupyter=True)
```



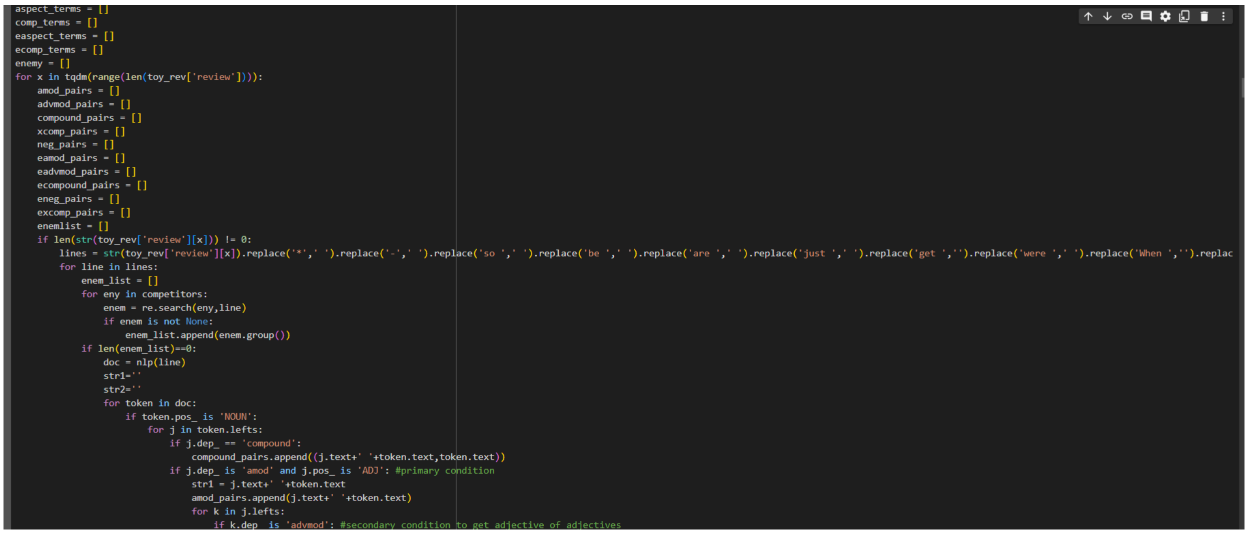
```
txt = 'We had a great experience at the restaurant, food was delicious, but the service was kinda bad'
doc = nlp(txt)
spacy.displacy.render(doc,style='dep',jupyter=True)
```



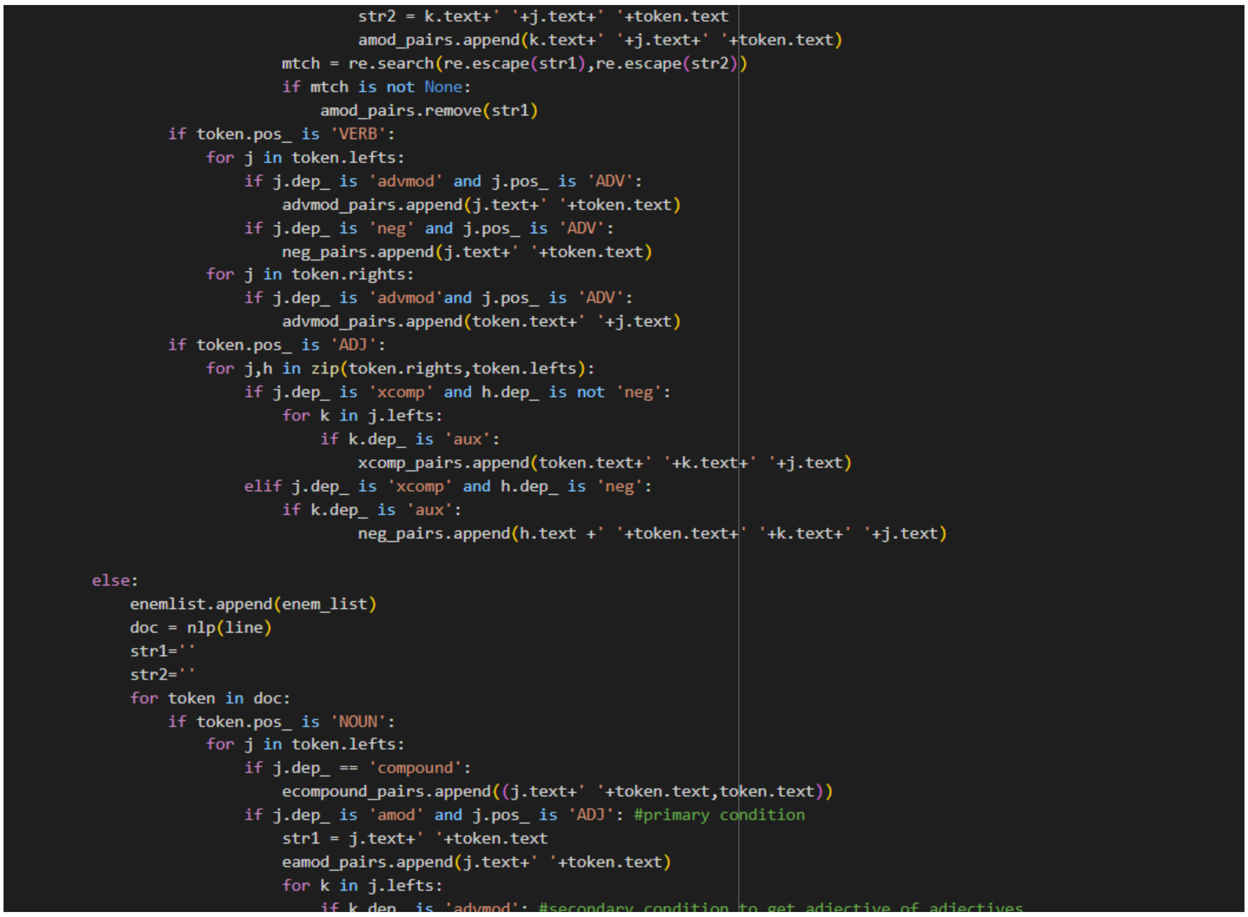
```
txt =  "wonderful to drive the camry "
doc = nlp(txt)
spacy.displacy.render(doc,style='dep',jupyter=True)
```



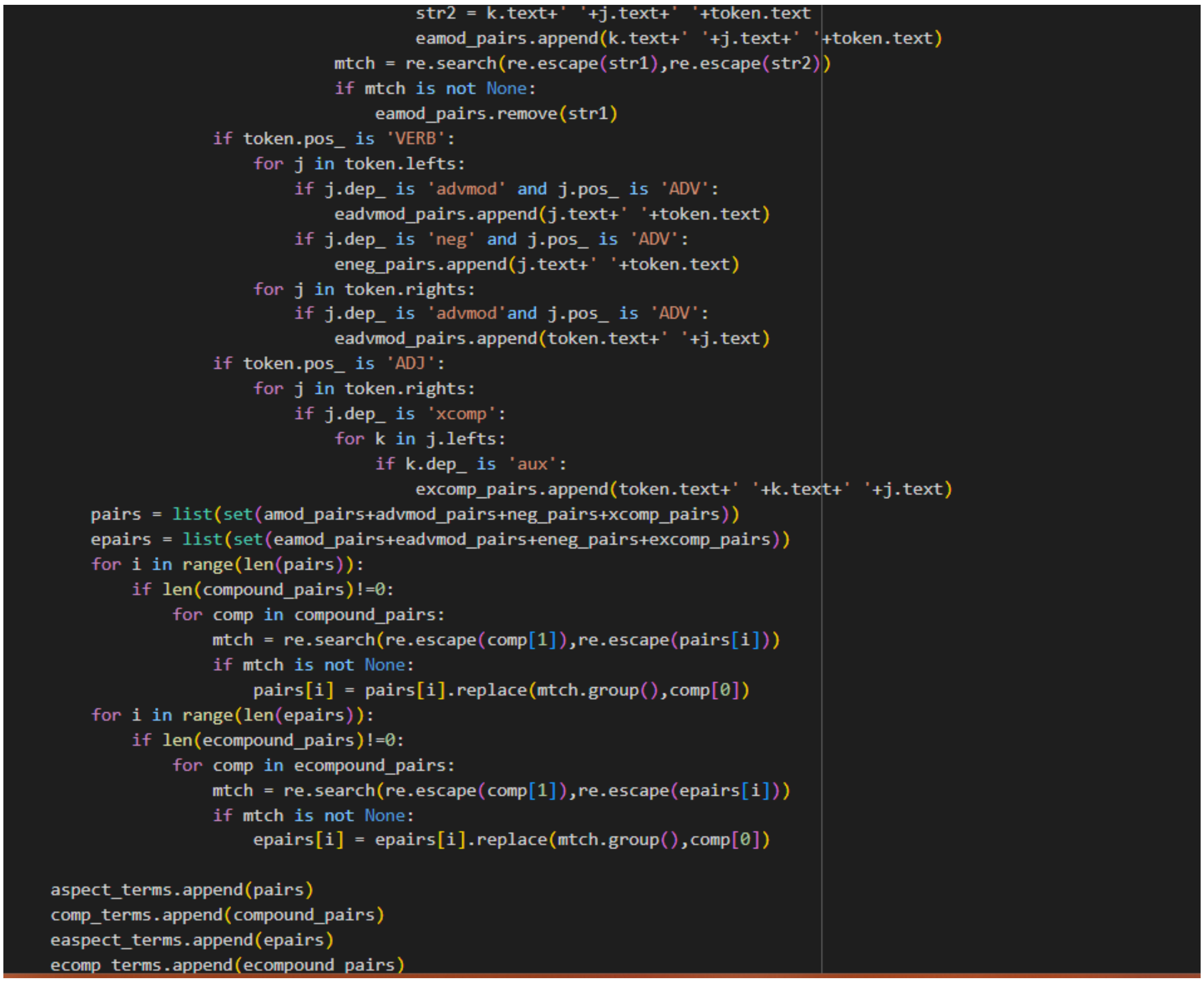
```
[ ]  competitors = ['Chevy','chevy','Ford','ford','Nissan','nissan','Honda','honda','Chevrolet','chevrolet','Volkswagen','volkswagen','benz','Benz','Mercedes','mercedes','subaru','Subaru','VW']
```



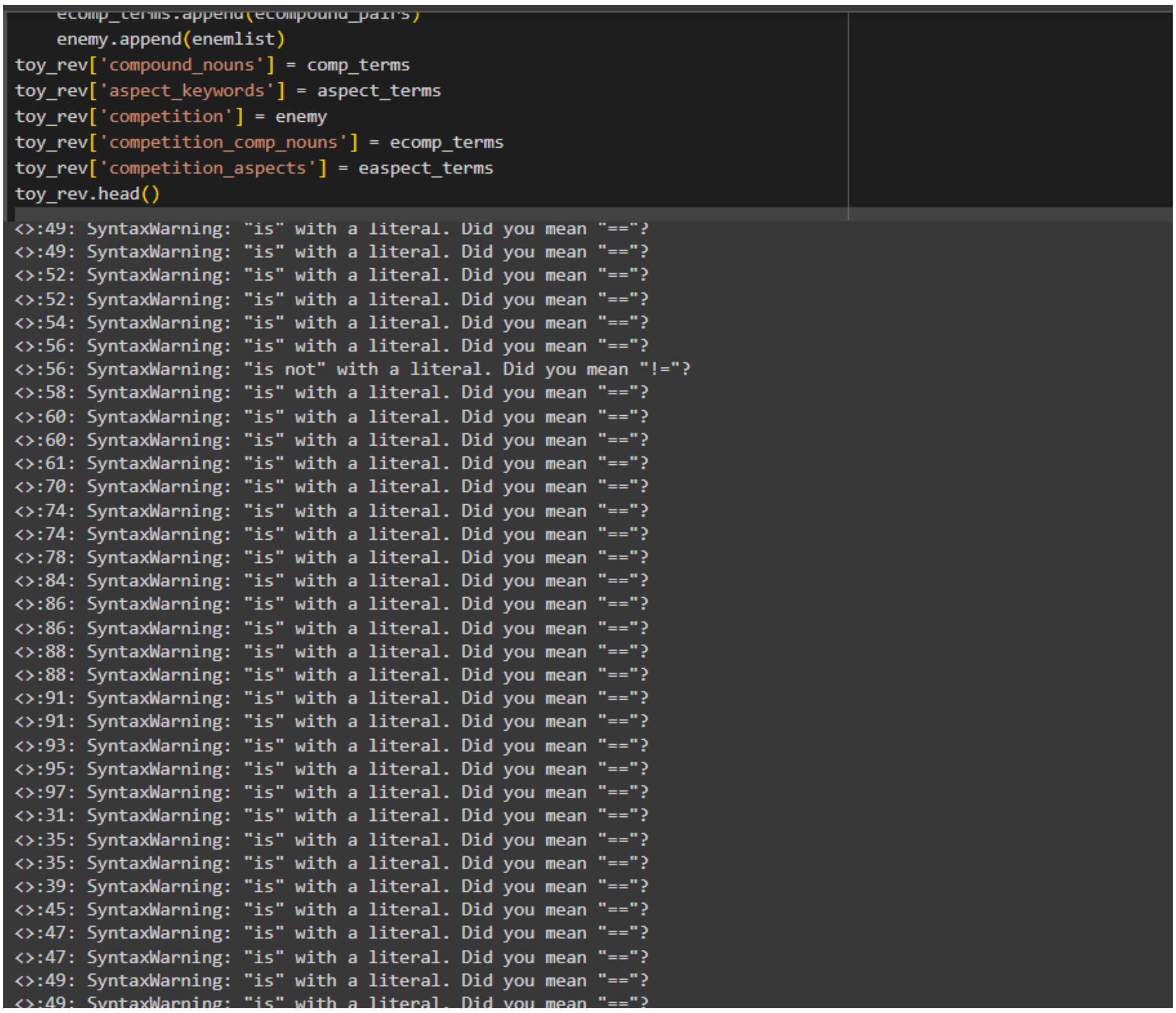
```python
aspect_terms = []
comp_terms = []
easpect_terms = []
ecomp_terms = []
enemy = []
for x in tqdm(range(len(toy_rev['review']))):
    amod_pairs = []
    advmod_pairs = []
    compound_pairs = []
    xcomp_pairs = []
    neg_pairs = []
    eamod_pairs = []
    eadvmod_pairs = []
    ecompound_pairs = []
    eneg_pairs = []
    excomp_pairs = []
    enemlist = []
    if len(str(toy_rev['review'][x])) != 0:
        lines = str(toy_rev['review'][x]).replace('*',' ').replace('-',' ').replace('so ',' ').replace('be ',' ').replace('are ',' ').replace('just ',' ').replace('get ','').replace('were ',' ').replace('When ','').replac
        for line in lines:
            enem_list = []
            for eny in competitors:
                enem = re.search(eny,line)
                if enem is not None:
                    enem_list.append(enem.group())
            if len(enem_list)==0:
                doc = nlp(line)
                str1=''
                str2=''
                for token in doc:
                    if token.pos_ is 'NOUN':
                        for j in token.lefts:
                            if j.dep_ == 'compound':
                                compound_pairs.append((j.text+' '+token.text,token.text))
                            if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary condition
                                str1 = j.text+' '+token.text
                                amod_pairs.append(j.text+' '+token.text)
                                for k in j.lefts:
                                    if k.dep_ is 'advmod': #secondary condition to get adjective of adjectives
```



```python
                                        str2 = k.text+' '+j.text+' '+token.text
                                        amod_pairs.append(k.text+' '+j.text+' '+token.text)
                                mtch = re.search(re.escape(str1),re.escape(str2))
                                if mtch is not None:
                                    amod_pairs.remove(str1)
                    if token.pos_ is 'VERB':
                        for j in token.lefts:
                            if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                                advmod_pairs.append(j.text+' '+token.text)
                            if j.dep_ is 'neg' and j.pos_ is 'ADV':
                                neg_pairs.append(j.text+' '+token.text)
                        for j in token.rights:
                            if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                                advmod_pairs.append(token.text+' '+j.text)
                    if token.pos_ is 'ADJ':
                        for j,h in zip(token.rights,token.lefts):
                            if j.dep_ is 'xcomp' and h.dep_ is not 'neg':
                                for k in j.lefts:
                                    if k.dep_ is 'aux':
                                        xcomp_pairs.append(token.text+' '+k.text+' '+j.text)
                            elif j.dep_ is 'xcomp' and h.dep_ is 'neg':
                                if k.dep_ is 'aux':
                                        neg_pairs.append(h.text +' '+token.text+' '+k.text+' '+j.text)

            else:
                enemlist.append(enem_list)
                doc = nlp(line)
                str1=''
                str2=''
                for token in doc:
                    if token.pos_ is 'NOUN':
                        for j in token.lefts:
                            if j.dep_ == 'compound':
                                ecompound_pairs.append((j.text+' '+token.text,token.text))
                            if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary condition
                                str1 = j.text+' '+token.text
                                eamod_pairs.append(j.text+' '+token.text)
                                for k in j.lefts:
                                    if k.dep_ is 'advmod': #secondary condition to get adjective of adjectives
```

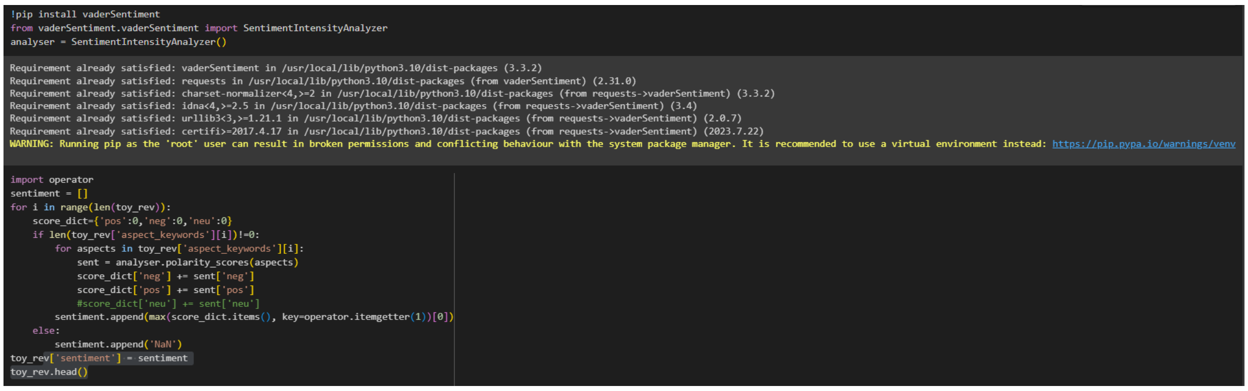


```
                                    str2 = k.text+' '+j.text+' '+token.text
                                    eamod_pairs.append(k.text+' '+j.text+' '+token.text)
                            mtch = re.search(re.escape(str1),re.escape(str2))
                            if mtch is not None:
                                eamod_pairs.remove(str1)
                if token.pos_ is 'VERB':
                    for j in token.lefts:
                        if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                            eadvmod_pairs.append(j.text+' '+token.text)
                        if j.dep_ is 'neg' and j.pos_ is 'ADV':
                            eneg_pairs.append(j.text+' '+token.text)
                    for j in token.rights:
                        if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                            eadvmod_pairs.append(token.text+' '+j.text)
                if token.pos_ is 'ADJ':
                    for j in token.rights:
                        if j.dep_ is 'xcomp':
                            for k in j.lefts:
                                if k.dep_ is 'aux':
                                    excomp_pairs.append(token.text+' '+k.text+' '+j.text)
    pairs = list(set(amod_pairs+advmod_pairs+neg_pairs+xcomp_pairs))
    epairs = list(set(eamod_pairs+eadvmod_pairs+eneg_pairs+excomp_pairs))
    for i in range(len(pairs)):
        if len(compound_pairs)!=0:
            for comp in compound_pairs:
                mtch = re.search(re.escape(comp[1]),re.escape(pairs[i]))
                if mtch is not None:
                    pairs[i] = pairs[i].replace(mtch.group(),comp[0])
    for i in range(len(epairs)):
        if len(ecompound_pairs)!=0:
            for comp in ecompound_pairs:
                mtch = re.search(re.escape(comp[1]),re.escape(epairs[i]))
                if mtch is not None:
                    epairs[i] = epairs[i].replace(mtch.group(),comp[0])

aspect_terms.append(pairs)
comp_terms.append(compound_pairs)
easpect_terms.append(epairs)
ecomp_terms.append(ecompound_pairs)
```



```
    ecomp_terms.append(ecompound_pairs)
    enemy.append(enemlist)
toy_rev['compound_nouns'] = comp_terms
toy_rev['aspect_keywords'] = aspect_terms
toy_rev['competition'] = enemy
toy_rev['competition_comp_nouns'] = ecomp_terms
toy_rev['competition_aspects'] = easpect_terms
toy_rev.head()
```

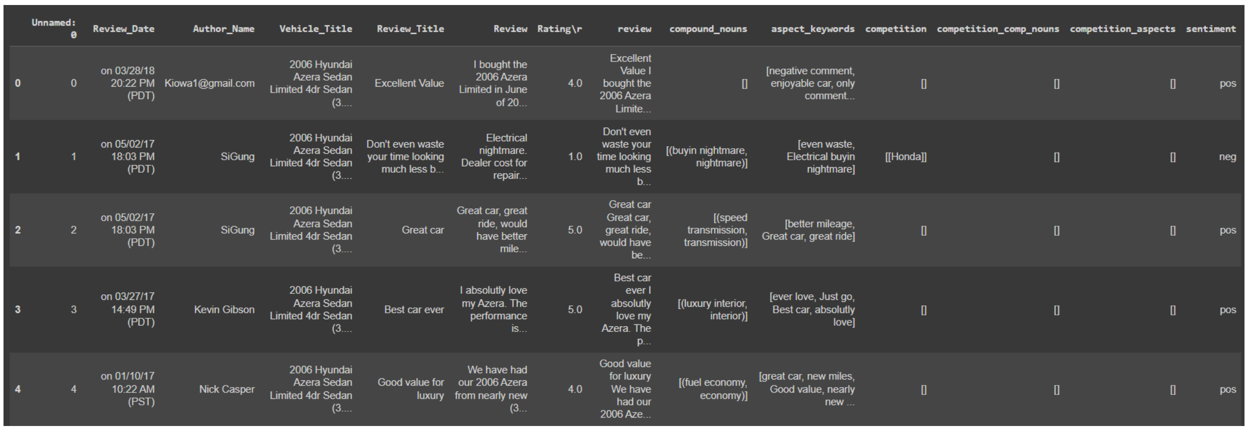
```
<>:49: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:49: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:52: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:52: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:54: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:56: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:56: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:58: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:60: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:60: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:61: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:70: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:74: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:74: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:78: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:84: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:86: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:86: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:88: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:88: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:91: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:91: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:93: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:95: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:97: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:31: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:35: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:35: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:39: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:45: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:47: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:47: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:49: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:49: SyntaxWarning: "is" with a literal. Did you mean "=="?
```



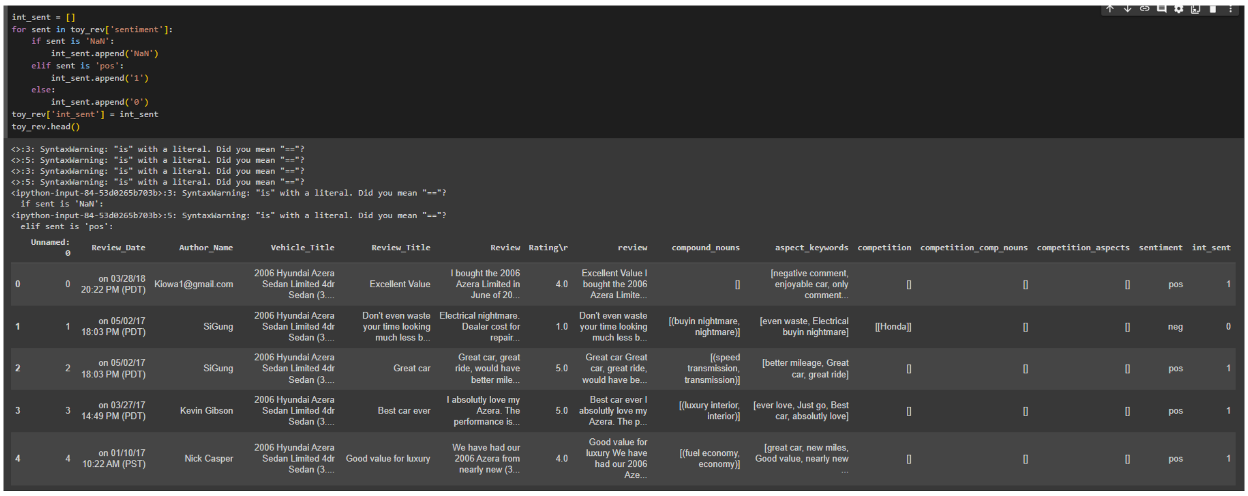
```
!pip install vaderSentiment
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()
```

```
Requirement already satisfied: vaderSentiment in /usr/local/lib/python3.10/dist-packages (3.3.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from vaderSentiment) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2023.7.22)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

```
import operator
sentiment = []
for i in range(len(toy_rev)):
    score_dict={'pos':0,'neg':0,'neu':0}
    if len(toy_rev['aspect_keywords'][i])!=0:
        for aspects in toy_rev['aspect_keywords'][i]:
            sent = analyser.polarity_scores(aspects)
            score_dict['neg'] += sent['neg']
            score_dict['pos'] += sent['pos']
            #score_dict['neu'] += sent['neu']
        sentiment.append(max(score_dict.items(), key=operator.itemgetter(1))[0])
    else:
        sentiment.append('NaN')
toy_rev['sentiment'] = sentiment
toy_rev.head()
```



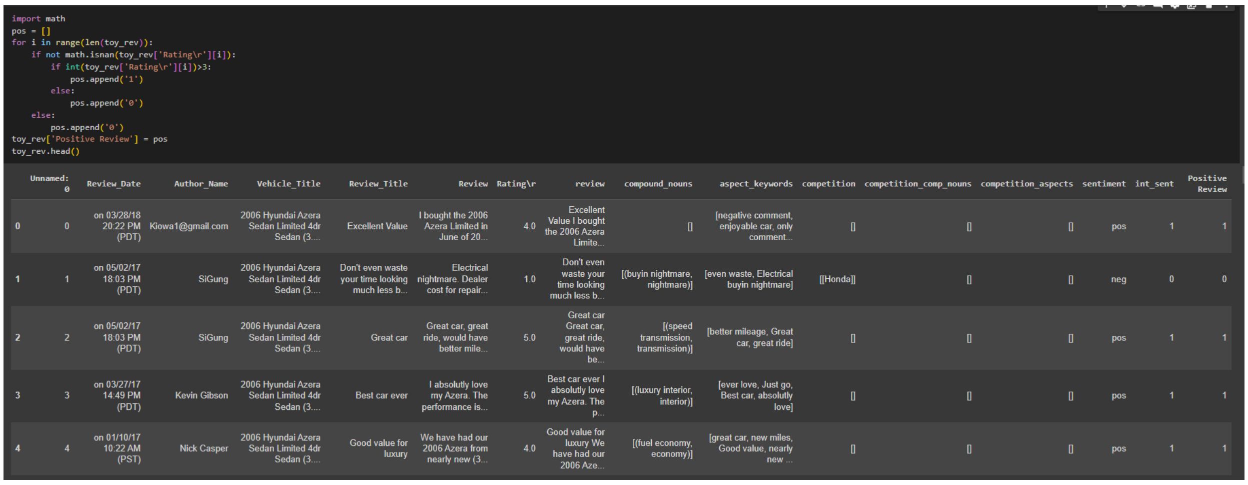
	Unnamed: 0	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating\r	review	compound_nouns	aspect_keywords	competition	competition_comp_nouns	competition_aspects	sentiment
0	0	on 03/28/18 20:22 PM (PDT)	Kiowa1@gmail.com	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Excellent Value	I bought the 2006 Azera Limited in June of 20...	4.0	Excellent Value I bought the 2006 Azera Limite...	[]	[negative comment, enjoyable car, only comment...	[]	[]	[]	pos
1	1	on 05/02/17 18:03 PM (PDT)	SiGung	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Don't even waste your time looking much less b...	Electrical nightmare. Dealer cost for repair...	1.0	Don't even waste your time looking much less b...	[(buyin nightmare, nightmare)]	[even waste, Electrical buyin nightmare]	[[Honda]]	[]	[]	neg
2	2	on 05/02/17 18:03 PM (PDT)	SiGung	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Great car	Great car, great ride, would have better mile...	5.0	Great car Great car, great ride, would have be...	[(speed transmission, transmission)]	[better mileage, Great car, great ride]	[]	[]	[]	pos
3	3	on 03/27/17 14:49 PM (PDT)	Kevin Gibson	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Best car ever	I absolutly love my Azera. The performance is...	5.0	Best car ever I absolutly love my Azera. The p...	[(luxury interior, interior)]	[ever love, Just go, Best car, absolutly love]	[]	[]	[]	pos
4	4	on 01/10/17 10:22 AM (PST)	Nick Casper	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Good value for luxury	We have had our 2006 Azera from nearly new (3...	4.0	Good value for luxury We have had our 2006 Aze...	[(fuel economy, economy)]	[great car, new miles, Good value, nearly new ...	[]	[]	[]	pos



```
int_sent = []
for sent in toy_rev['sentiment']:
    if sent is 'NaN':
        int_sent.append('NaN')
    elif sent is 'pos':
        int_sent.append('1')
    else:
        int_sent.append('0')
toy_rev['int_sent'] = int_sent
toy_rev.head()
```

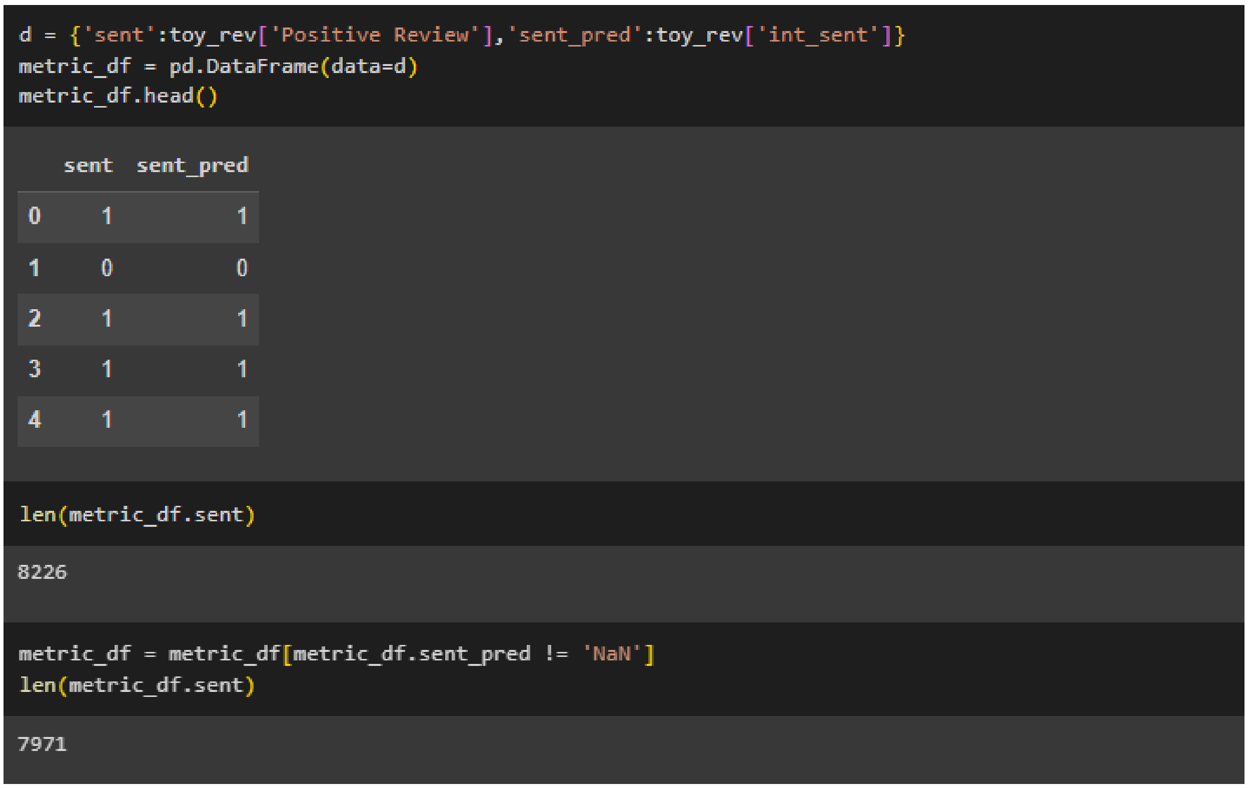
```
<>:3: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:3: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
<ipython-input-84-53d0265b703b>:3: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if sent is 'NaN':
<ipython-input-84-53d0265b703b>:5: SyntaxWarning: "is" with a literal. Did you mean "=="?
  elif sent is 'pos':
```

	Unnamed: 0	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating\r	review	compound_nouns	aspect_keywords	competition	competition_comp_nouns	competition_aspects	sentiment	int_sent
0	0	on 03/28/18 20:22 PM (PDT)	Kiowa1@gmail.com	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Excellent Value	I bought the 2006 Azera Limited in June of 20...	4.0	Excellent Value I bought the 2006 Azera Limite...	[]	[negative comment, enjoyable car, only comment...	[]	[]	[]	pos	1
1	1	on 05/02/17 18:03 PM (PDT)	SiGung	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Don't even waste your time looking much less b...	Electrical nightmare. Dealer cost for repair...	1.0	Don't even waste your time looking much less b...	[(buyin nightmare, nightmare)]	[even waste, Electrical buyin nightmare]	[[Honda]]	[]	[]	neg	0
2	2	on 05/02/17 18:03 PM (PDT)	SiGung	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Great car	Great car, great ride, would have better mile...	5.0	Great car Great car, great ride, would have be...	[(speed transmission, transmission)]	[better mileage, Great car, great ride]	[]	[]	[]	pos	1
3	3	on 03/27/17 14:49 PM (PDT)	Kevin Gibson	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Best car ever	I absolutly love my Azera. The performance is...	5.0	Best car ever I absolutly love my Azera. The p...	[(luxury interior, interior)]	[ever love, Just go, Best car, absolutly love]	[]	[]	[]	pos	1
4	4	on 01/10/17 10:22 AM (PST)	Nick Casper	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Good value for luxury	We have had our 2006 Azera from nearly new (3...	4.0	Good value for luxury We have had our 2006 Aze...	[(fuel economy, economy)]	[great car, new miles, Good value, nearly new ...	[]	[]	[]	pos	1



```python
import math
pos = []
for i in range(len(toy_rev)):
    if not math.isnan(toy_rev['Rating\r'][i]):
        if int(toy_rev['Rating\r'][i])>3:
            pos.append('1')
        else:
            pos.append('0')
    else:
        pos.append('0')
toy_rev['Positive Review'] = pos
toy_rev.head()
```

	Unnamed: 0	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating\r	review	compound_nouns	aspect_keywords	competition	competition_comp_nouns	competition_aspects	sentiment	int_sent	Positive Review
0	0	on 03/28/18 20:22 PM (PDT)	Kiowa1@gmail.com	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Excellent Value	I bought the 2006 Azera Limited in June of 20...	4.0	Excellent Value I bought the 2006 Azera Limite...	[]	[negative comment, enjoyable car, only comment...	[]	[]	[]	pos	1	1
1	1	on 05/02/17 18:03 PM (PDT)	SiGung	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Don't even waste your time looking much less b...	Electrical nightmare. Dealer cost for repair...	1.0	Don't even waste your time looking much less b...	[(buyin nightmare, nightmare)]	[even waste, Electrical buyin nightmare]	[[Honda]]	[]	[]	neg	0	0
2	2	on 05/02/17 18:03 PM (PDT)	SiGung	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Great car	Great car, great ride, would have better mile...	5.0	Great car Great car, great ride, would have be...	[(speed transmission, transmission)]	[better mileage, Great car, great ride]	[]	[]	[]	pos	1	1
3	3	on 03/27/17 14:49 PM (PDT)	Kevin Gibson	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Best car ever	I absolutly love my Azera. The performance is...	5.0	Best car ever I absolutly love my Azera. The p...	[(luxury interior, interior)]	[ever love, Just go, Best car, absolutly love]	[]	[]	[]	pos	1	1
4	4	on 01/10/17 10:22 AM (PST)	Nick Casper	2006 Hyundai Azera Sedan Limited 4dr Sedan (3....	Good value for luxury	We have had our 2006 Azera from nearly new (3...	4.0	Good value for luxury We have had our 2006 Aze...	[(fuel economy, economy)]	[great car, new miles, Good value, nearly new ...	[]	[]	[]	pos	1	1



```python
d = {'sent':toy_rev['Positive Review'],'sent_pred':toy_rev['int_sent']}
metric_df = pd.DataFrame(data=d)
metric_df.head()
```

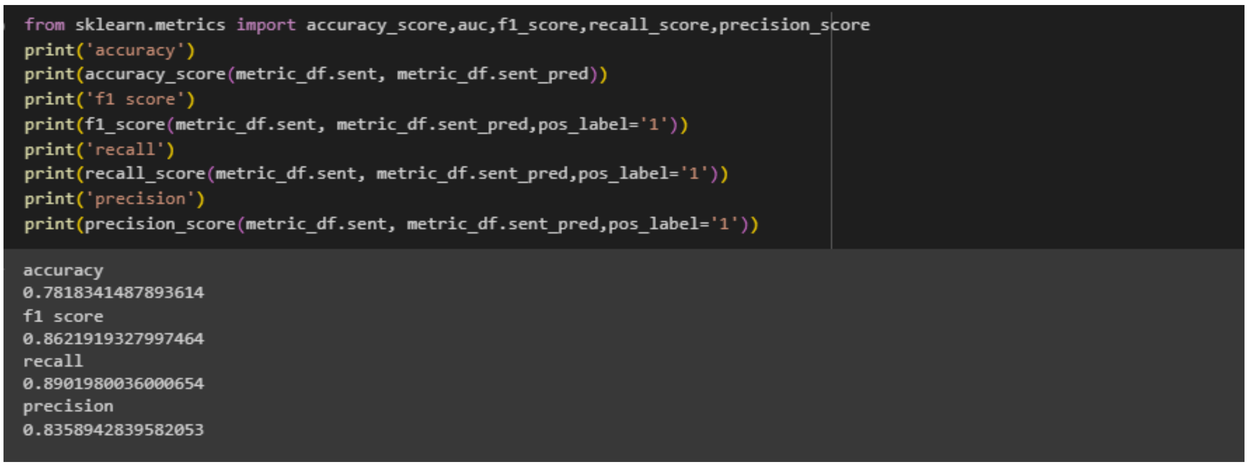
	sent	sent_pred
0	1	1
1	0	0
2	1	1
3	1	1
4	1	1

```python
len(metric_df.sent)
```

```
8226
```

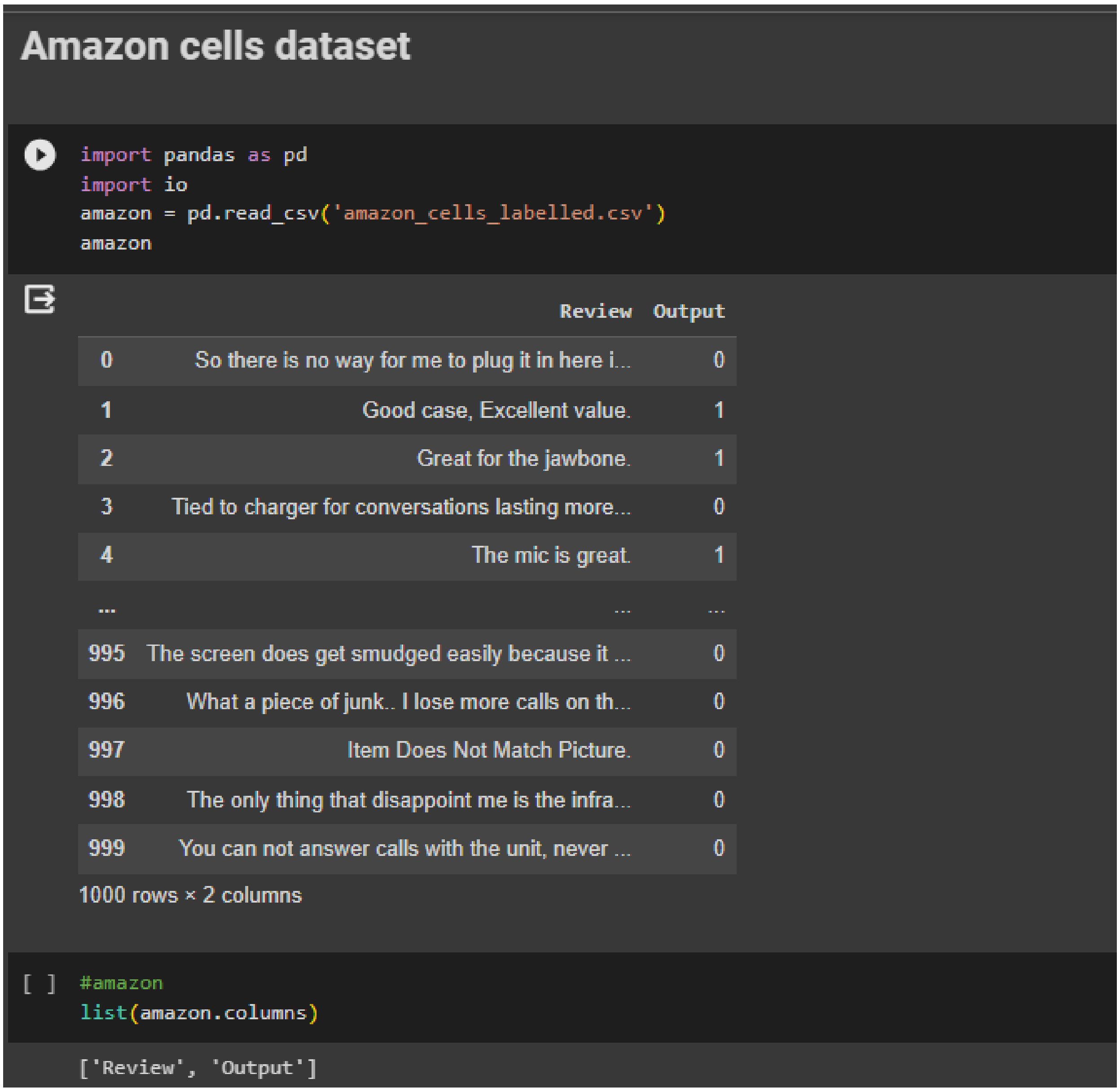
```python
metric_df = metric_df[metric_df.sent_pred != 'NaN']
len(metric_df.sent)
```

```
7971
```



```
from sklearn.metrics import accuracy_score,auc,f1_score,recall_score,precision_score
print('accuracy')
print(accuracy_score(metric_df.sent, metric_df.sent_pred))
print('f1 score')
print(f1_score(metric_df.sent, metric_df.sent_pred,pos_label='1'))
print('recall')
print(recall_score(metric_df.sent, metric_df.sent_pred,pos_label='1'))
print('precision')
print(precision_score(metric_df.sent, metric_df.sent_pred,pos_label='1'))
```

```
accuracy
0.7818341487893614
f1 score
0.8621919327997464
recall
0.8901980036000654
precision
0.8358942839582053
```



Amazon cells dataset

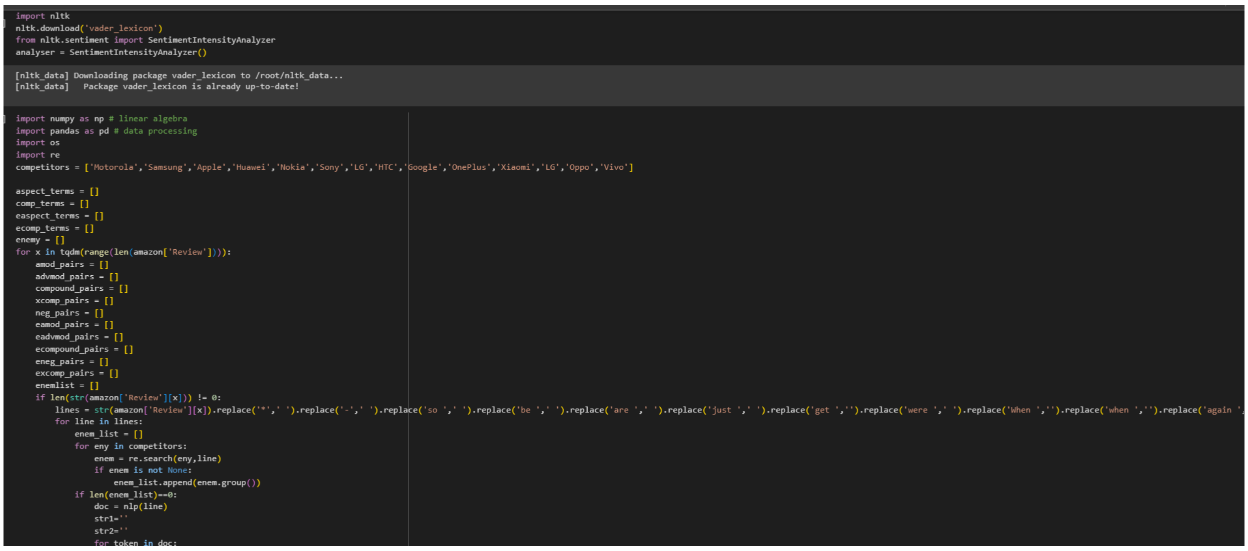
```
import pandas as pd
import io
amazon = pd.read_csv('amazon_cells_labelled.csv')
amazon
```

	Review	Output
0	So there is no way for me to plug it in here i...	0
1	Good case, Excellent value.	1
2	Great for the jawbone.	1
3	Tied to charger for conversations lasting more...	0
4	The mic is great.	1
...	...	...
995	The screen does get smudged easily because it ...	0
996	What a piece of junk.. I lose more calls on th...	0
997	Item Does Not Match Picture.	0
998	The only thing that disappoint me is the infra...	0
999	You can not answer calls with the unit, never ...	0

1000 rows × 2 columns

```
[ ] #amazon
    list(amazon.columns)
```

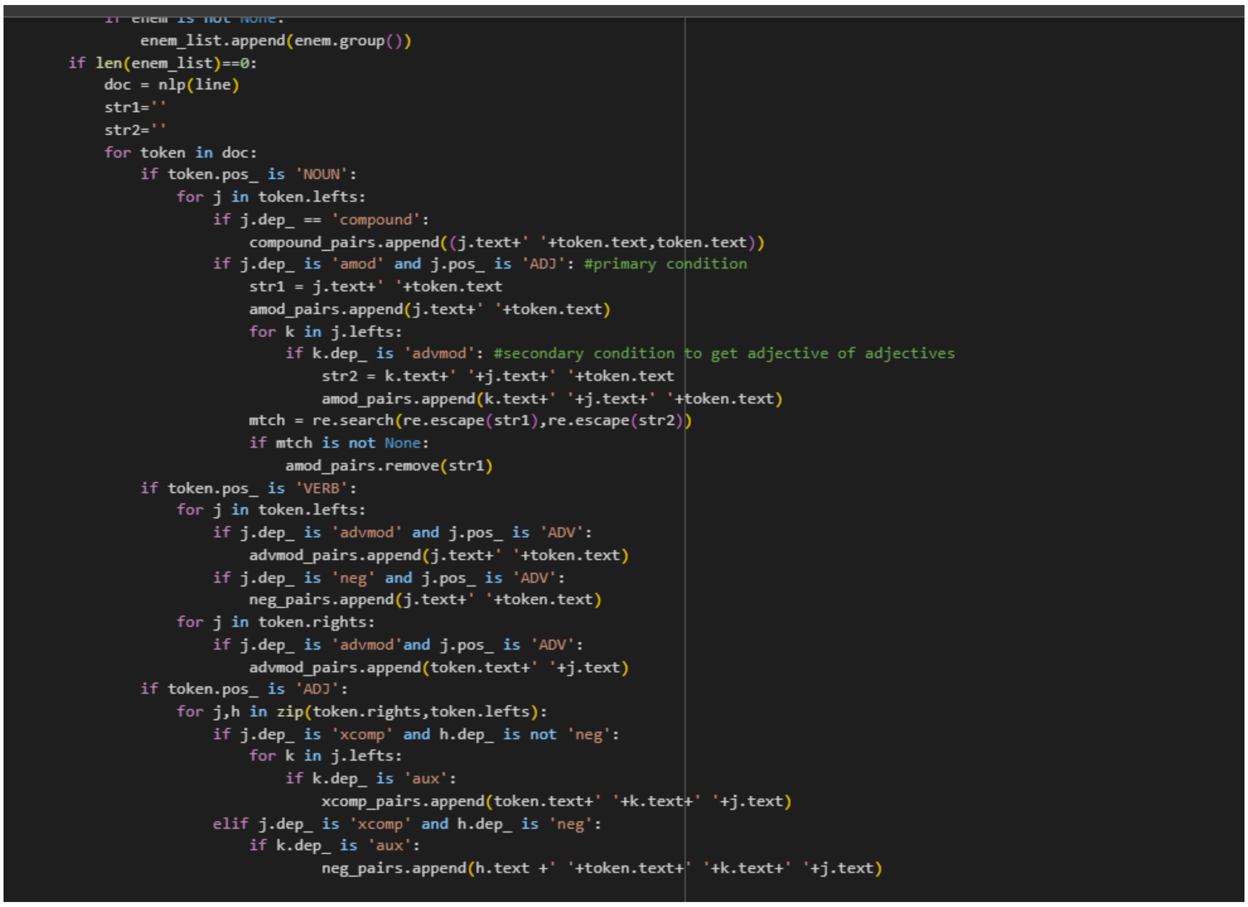
```
['Review', 'Output']
```



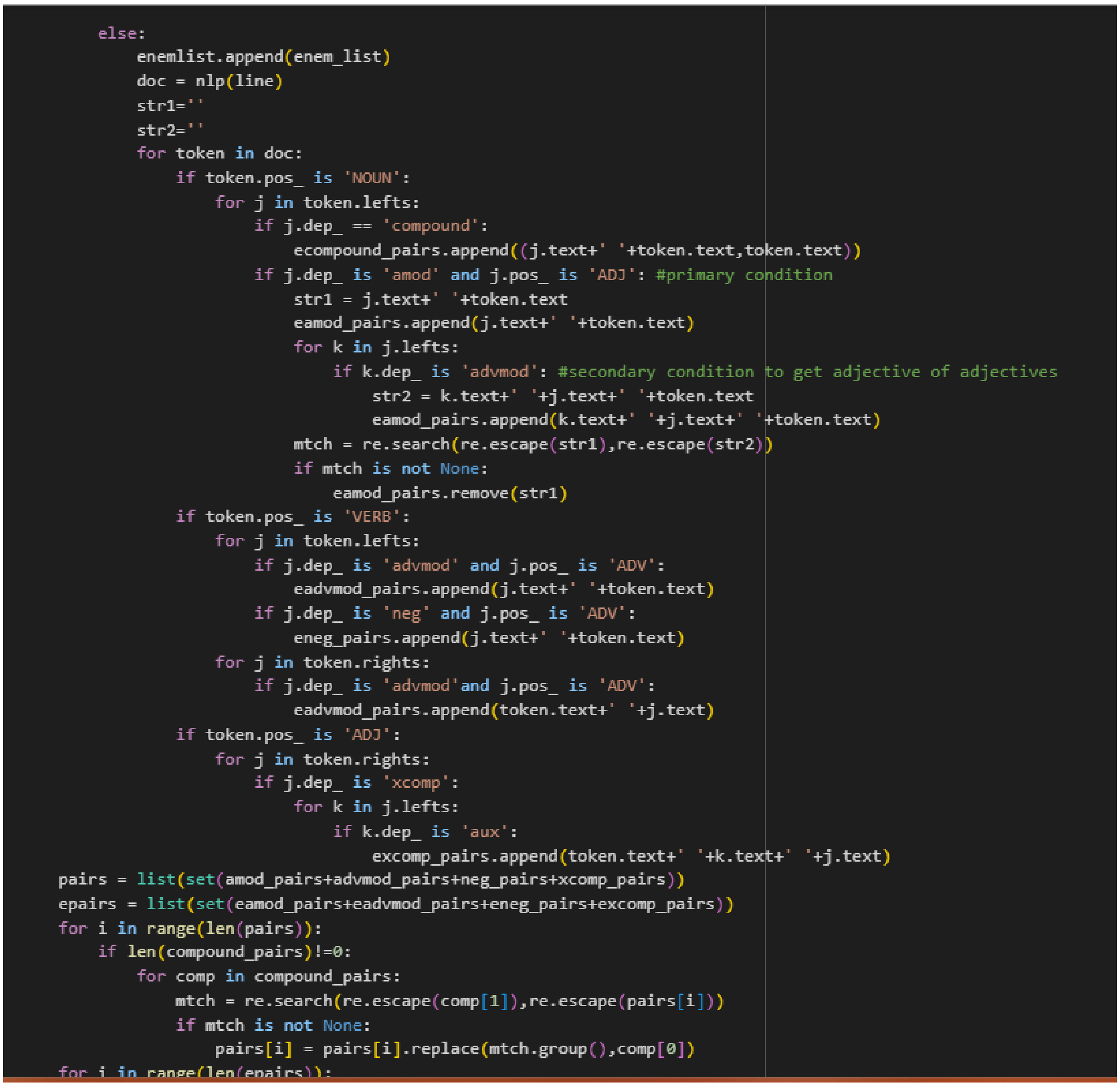
```
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

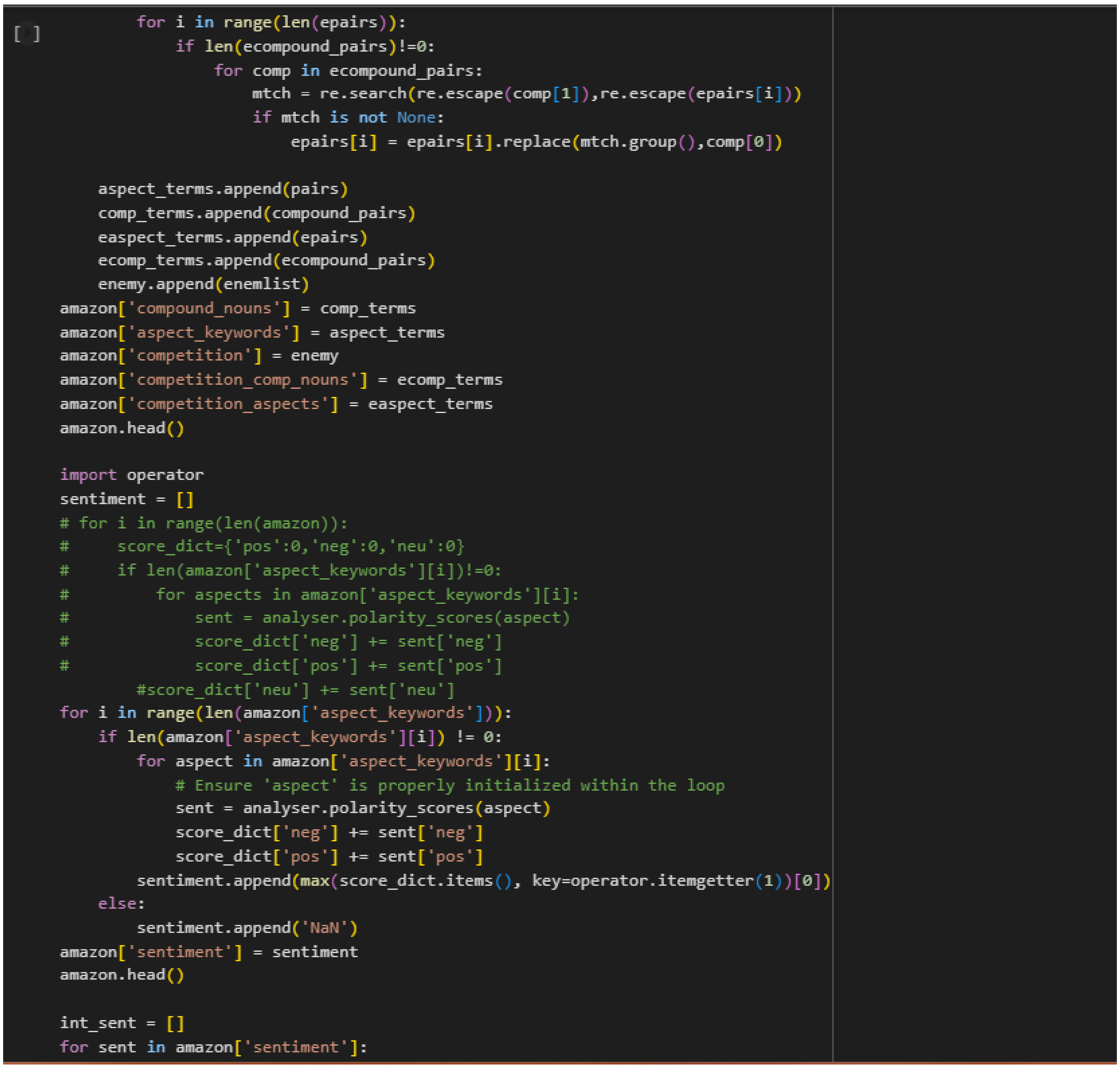
```
import numpy as np # linear algebra
import pandas as pd # data processing
import os
import re
competitors = ['Motorola','Samsung','Apple','Huawei','Nokia','Sony','LG','HTC','Google','OnePlus','Xiaomi','LG','Oppo','Vivo']

aspect_terms = []
comp_terms = []
easpect_terms = []
ecomp_terms = []
enemy = []
for x in tqdm(range(len(amazon['Review']))):
    amod_pairs = []
    advmod_pairs = []
    compound_pairs = []
    xcomp_pairs = []
    neg_pairs = []
    eamod_pairs = []
    eadvmod_pairs = []
    ecompound_pairs = []
    eneg_pairs = []
    excomp_pairs = []
    enemlist = []
    if len(str(amazon['Review'][x])) != 0:
        lines = str(amazon['Review'][x]).replace('*',' ').replace('-',' ').replace('so ',' ').replace('be ',' ').replace('are ',' ').replace('just ',' ').replace('get ','').replace('were ',' ').replace('When ','').replace('when ','').replace('again '
        for line in lines:
            enem_list = []
            for eny in competitors:
                enem = re.search(eny,line)
                if enem is not None:
                    enem_list.append(enem.group())
            if len(enem_list)==0:
                doc = nlp(line)
                str1=''
                str2=''
                for token in doc:
```



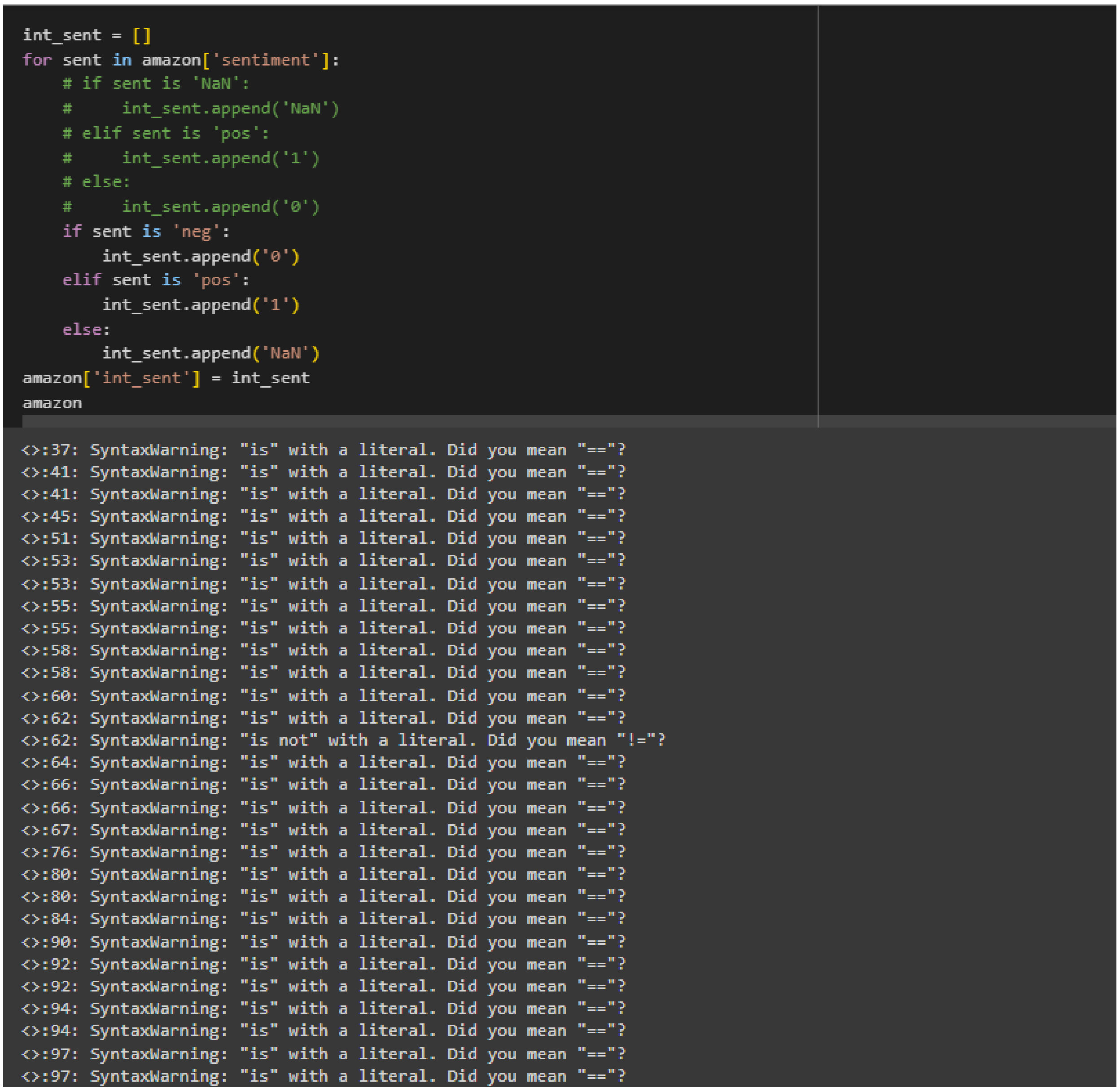
```
                if enem is not None:
                    enem_list.append(enem.group())
            if len(enem_list)==0:
                doc = nlp(line)
                str1=''
                str2=''
                for token in doc:
                    if token.pos_ is 'NOUN':
                        for j in token.lefts:
                            if j.dep_ == 'compound':
                                compound_pairs.append((j.text+' '+token.text,token.text))
                            if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary condition
                                str1 = j.text+' '+token.text
                                amod_pairs.append(j.text+' '+token.text)
                                for k in j.lefts:
                                    if k.dep_ is 'advmod': #secondary condition to get adjective of adjectives
                                        str2 = k.text+' '+j.text+' '+token.text
                                        amod_pairs.append(k.text+' '+j.text+' '+token.text)
                                mtch = re.search(re.escape(str1),re.escape(str2))
                                if mtch is not None:
                                    amod_pairs.remove(str1)
                    if token.pos_ is 'VERB':
                        for j in token.lefts:
                            if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                                advmod_pairs.append(j.text+' '+token.text)
                            if j.dep_ is 'neg' and j.pos_ is 'ADV':
                                neg_pairs.append(j.text+' '+token.text)
                        for j in token.rights:
                            if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                                advmod_pairs.append(token.text+' '+j.text)
                    if token.pos_ is 'ADJ':
                        for j,h in zip(token.rights,token.lefts):
                            if j.dep_ is 'xcomp' and h.dep_ is not 'neg':
                                for k in j.lefts:
                                    if k.dep_ is 'aux':
                                        xcomp_pairs.append(token.text+' '+k.text+' '+j.text)
                            elif j.dep_ is 'xcomp' and h.dep_ is 'neg':
                                if k.dep_ is 'aux':
                                        neg_pairs.append(h.text +' '+token.text+' '+k.text+' '+j.text)
```



```
    else:
        enemlist.append(enem_list)
        doc = nlp(line)
        str1=''
        str2=''
        for token in doc:
            if token.pos_ is 'NOUN':
                for j in token.lefts:
                    if j.dep_ == 'compound':
                        ecompound_pairs.append((j.text+' '+token.text,token.text))
                    if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary condition
                        str1 = j.text+' '+token.text
                        eamod_pairs.append(j.text+' '+token.text)
                        for k in j.lefts:
                            if k.dep_ is 'advmod': #secondary condition to get adjective of adjectives
                                str2 = k.text+' '+j.text+' '+token.text
                                eamod_pairs.append(k.text+' '+j.text+' '+token.text)
                        mtch = re.search(re.escape(str1),re.escape(str2))
                        if mtch is not None:
                            eamod_pairs.remove(str1)
            if token.pos_ is 'VERB':
                for j in token.lefts:
                    if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                        eadvmod_pairs.append(j.text+' '+token.text)
                    if j.dep_ is 'neg' and j.pos_ is 'ADV':
                        eneg_pairs.append(j.text+' '+token.text)
                for j in token.rights:
                    if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                        eadvmod_pairs.append(token.text+' '+j.text)
            if token.pos_ is 'ADJ':
                for j in token.rights:
                    if j.dep_ is 'xcomp':
                        for k in j.lefts:
                            if k.dep_ is 'aux':
                                excomp_pairs.append(token.text+' '+k.text+' '+j.text)
pairs = list(set(amod_pairs+advmod_pairs+neg_pairs+xcomp_pairs))
epairs = list(set(eamod_pairs+eadvmod_pairs+eneg_pairs+excomp_pairs))
for i in range(len(pairs)):
    if len(compound_pairs)!=0:
        for comp in compound_pairs:
            mtch = re.search(re.escape(comp[1]),re.escape(pairs[i]))
            if mtch is not None:
                pairs[i] = pairs[i].replace(mtch.group(),comp[0])
for i in range(len(epairs)):
```

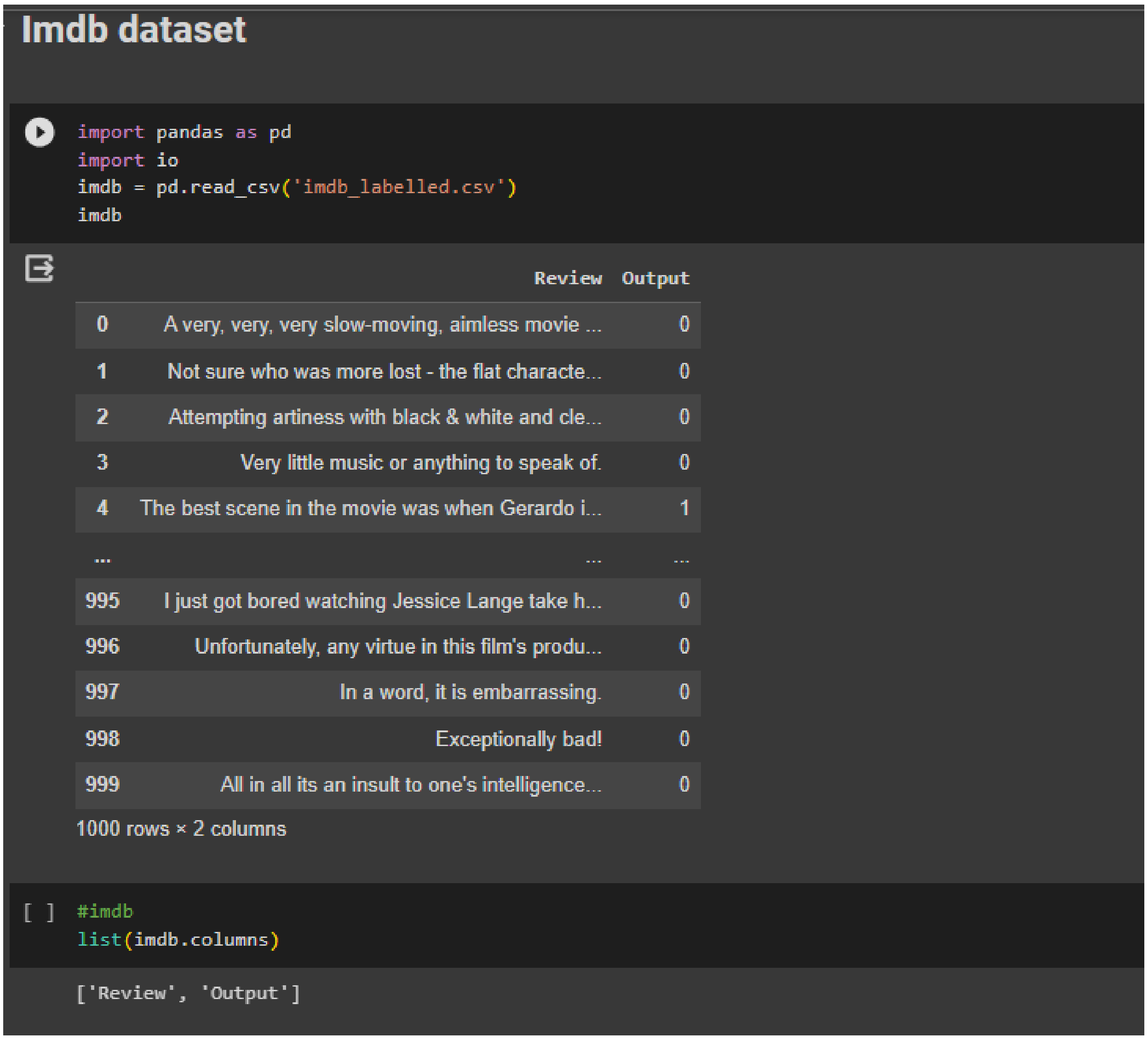


```
[ ]         for i in range(len(epairs)):
                if len(ecompound_pairs)!=0:
                    for comp in ecompound_pairs:
                        mtch = re.search(re.escape(comp[1]),re.escape(epairs[i]))
                        if mtch is not None:
                            epairs[i] = epairs[i].replace(mtch.group(),comp[0])

        aspect_terms.append(pairs)
        comp_terms.append(compound_pairs)
        easpect_terms.append(epairs)
        ecomp_terms.append(ecompound_pairs)
        enemy.append(enemlist)
    amazon['compound_nouns'] = comp_terms
    amazon['aspect_keywords'] = aspect_terms
    amazon['competition'] = enemy
    amazon['competition_comp_nouns'] = ecomp_terms
    amazon['competition_aspects'] = easpect_terms
    amazon.head()

    import operator
    sentiment = []
    # for i in range(len(amazon)):
    #     score_dict={'pos':0,'neg':0,'neu':0}
    #     if len(amazon['aspect_keywords'][i])!=0:
    #         for aspects in amazon['aspect_keywords'][i]:
    #             sent = analyser.polarity_scores(aspect)
    #             score_dict['neg'] += sent['neg']
    #             score_dict['pos'] += sent['pos']
            #score_dict['neu'] += sent['neu']
    for i in range(len(amazon['aspect_keywords'])):
        if len(amazon['aspect_keywords'][i]) != 0:
            for aspect in amazon['aspect_keywords'][i]:
                # Ensure 'aspect' is properly initialized within the loop
                sent = analyser.polarity_scores(aspect)
                score_dict['neg'] += sent['neg']
                score_dict['pos'] += sent['pos']
            sentiment.append(max(score_dict.items(), key=operator.itemgetter(1))[0])
        else:
            sentiment.append('NaN')
    amazon['sentiment'] = sentiment
    amazon.head()

    int_sent = []
    for sent in amazon['sentiment']:
```



```python
int_sent = []
for sent in amazon['sentiment']:
    # if sent is 'NaN':
    #     int_sent.append('NaN')
    # elif sent is 'pos':
    #     int_sent.append('1')
    # else:
    #     int_sent.append('0')
    if sent is 'neg':
        int_sent.append('0')
    elif sent is 'pos':
        int_sent.append('1')
    else:
        int_sent.append('NaN')
amazon['int_sent'] = int_sent
amazon
```

```
<>:37: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:41: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:41: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:45: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:51: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:53: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:53: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:55: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:55: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:58: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:58: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:60: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:62: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:62: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:64: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:66: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:66: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:67: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:76: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:80: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:80: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:84: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:90: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:92: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:92: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:94: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:94: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:97: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:97: SyntaxWarning: "is" with a literal. Did you mean "=="?
```



Imdb dataset

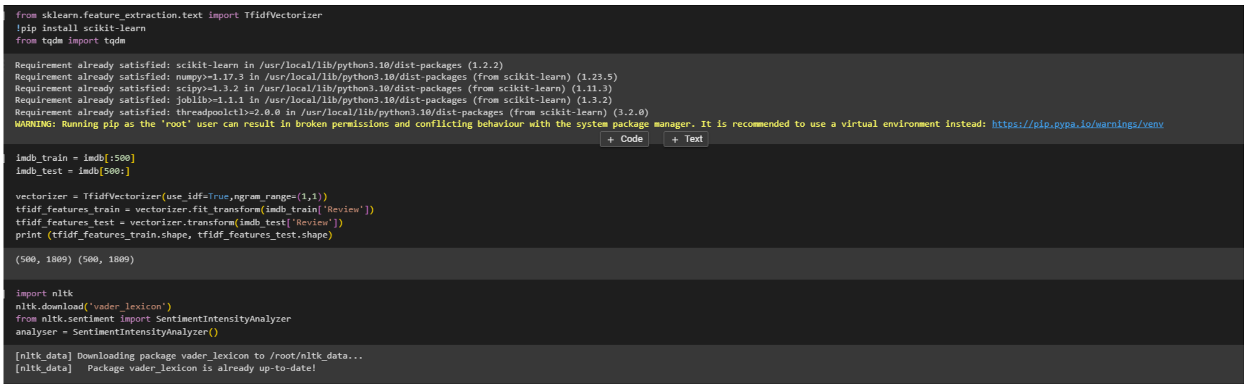
```python
import pandas as pd
import io
imdb = pd.read_csv('imdb_labelled.csv')
imdb
```

	Review	Output
0	A very, very, very slow-moving, aimless movie ...	0
1	Not sure who was more lost - the flat characte...	0
2	Attempting artiness with black & white and cle...	0
3	Very little music or anything to speak of.	0
4	The best scene in the movie was when Gerardo i...	1
...	...	...
995	I just got bored watching Jessice Lange take h...	0
996	Unfortunately, any virtue in this film's produ...	0
997	In a word, it is embarrassing.	0
998	Exceptionally bad!	0
999	All in all its an insult to one's intelligence...	0

1000 rows × 2 columns

```python
[ ] #imdb
    list(imdb.columns)
```

```
['Review', 'Output']
```



```python
from sklearn.feature_extraction.text import TfidfVectorizer
!pip install scikit-learn
from tqdm import tqdm
```

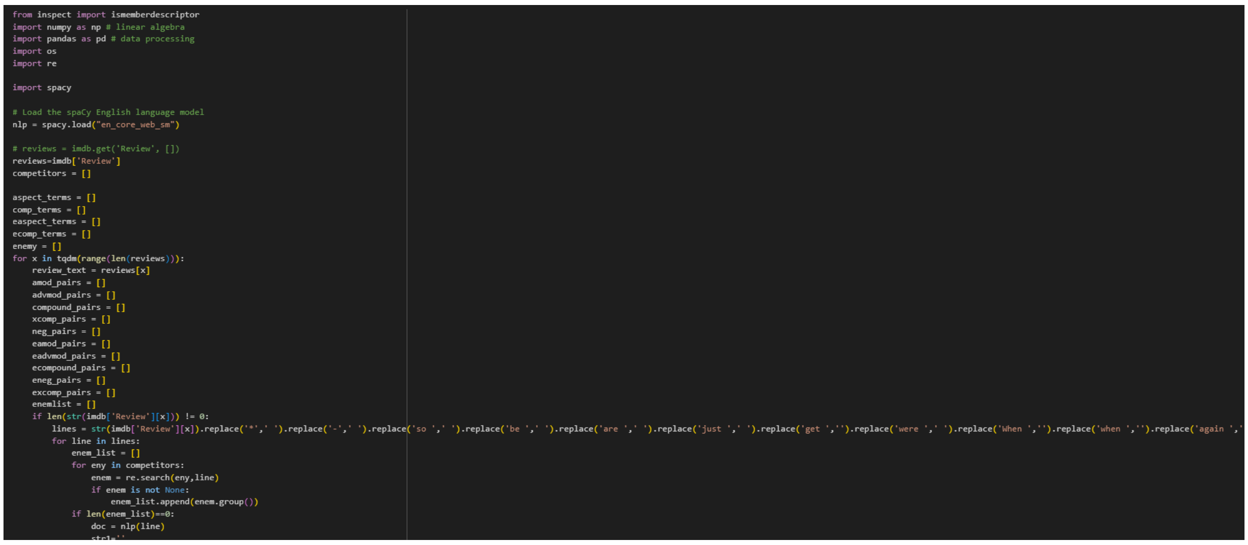
```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

```python
imdb_train = imdb[:500]
imdb_test = imdb[500:]

vectorizer = TfidfVectorizer(use_idf=True,ngram_range=(1,1))
tfidf_features_train = vectorizer.fit_transform(imdb_train['Review'])
tfidf_features_test = vectorizer.transform(imdb_test['Review'])
print (tfidf_features_train.shape, tfidf_features_test.shape)
```

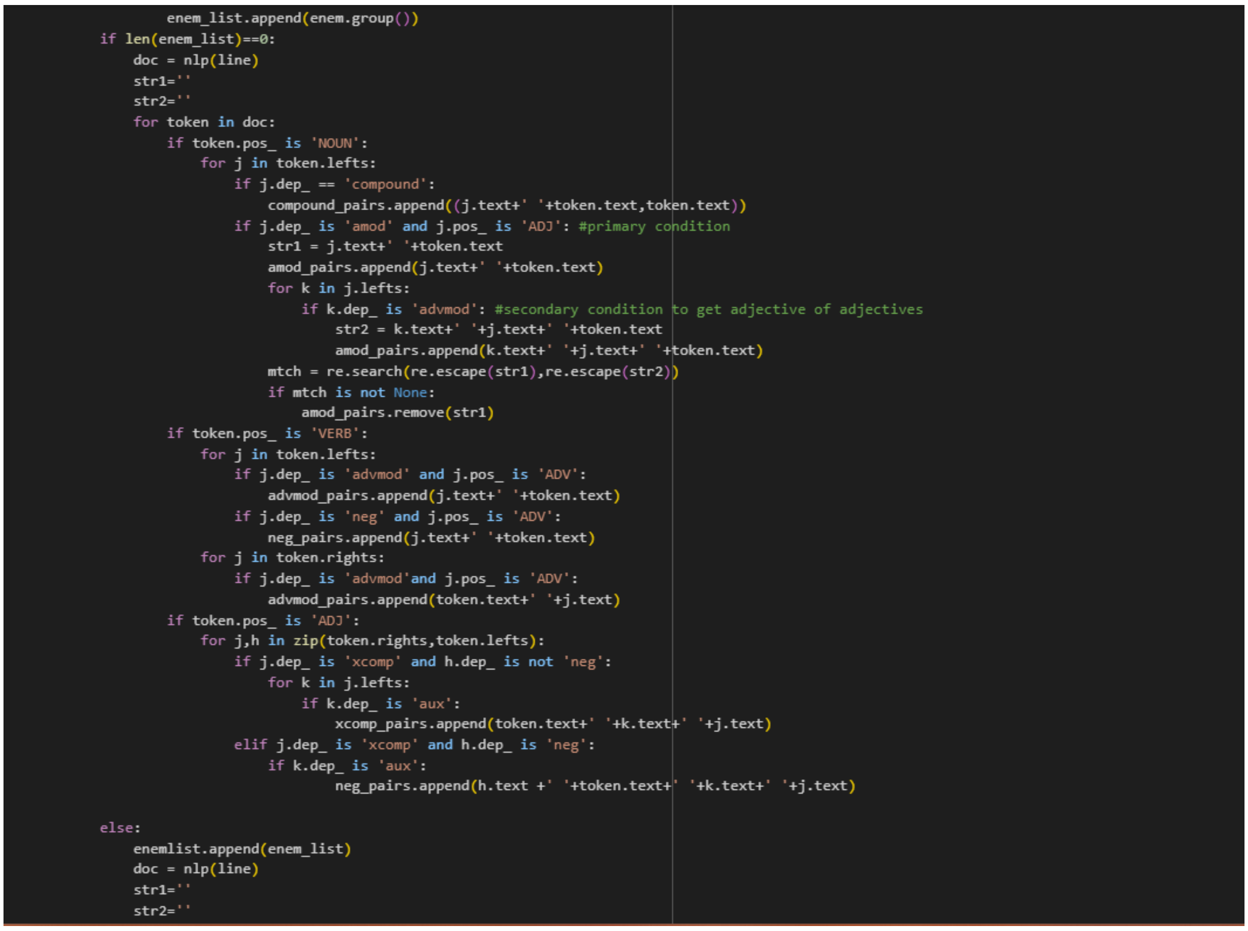
```
(500, 1809) (500, 1809)
```

```python
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()
```

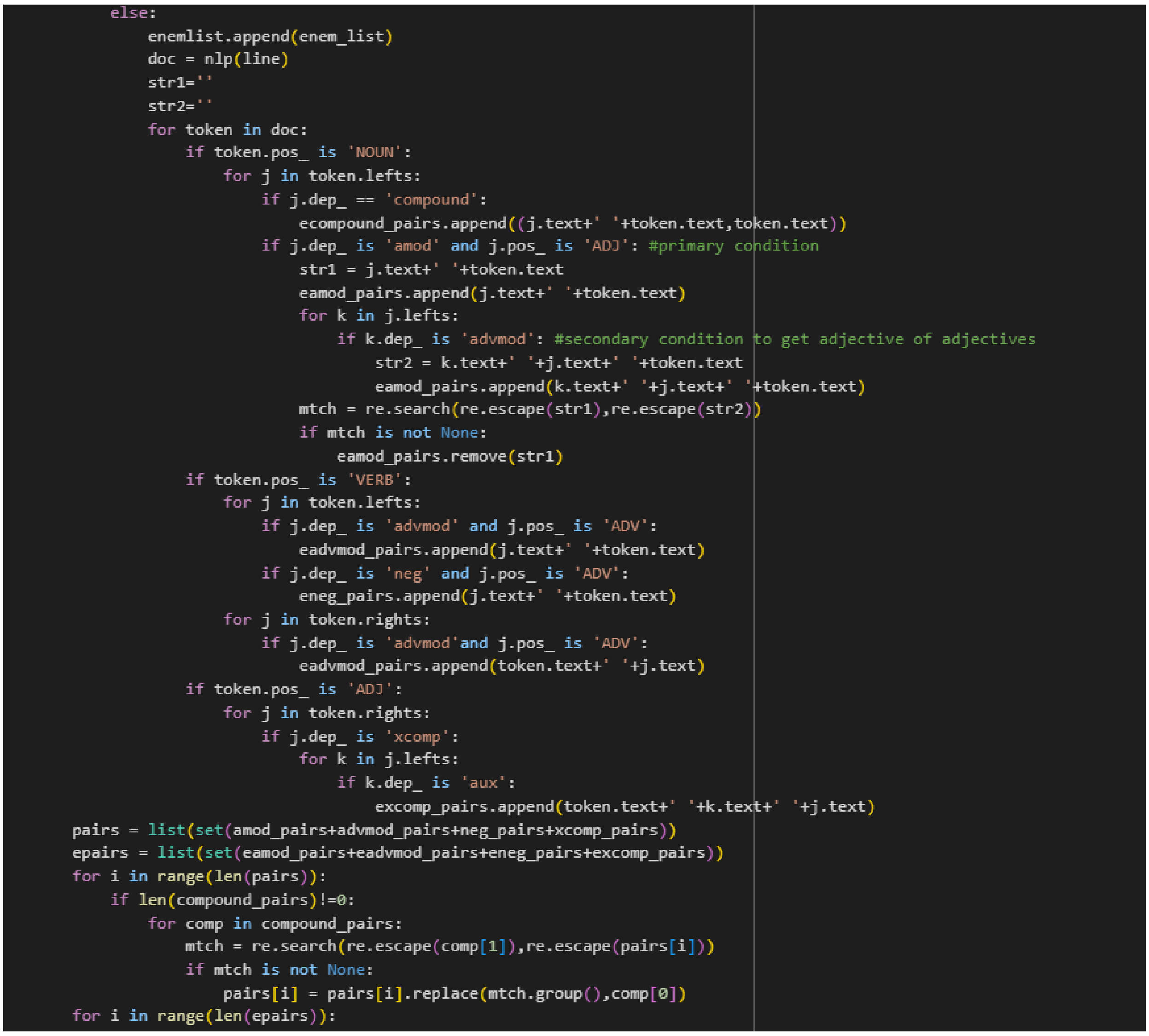
```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```



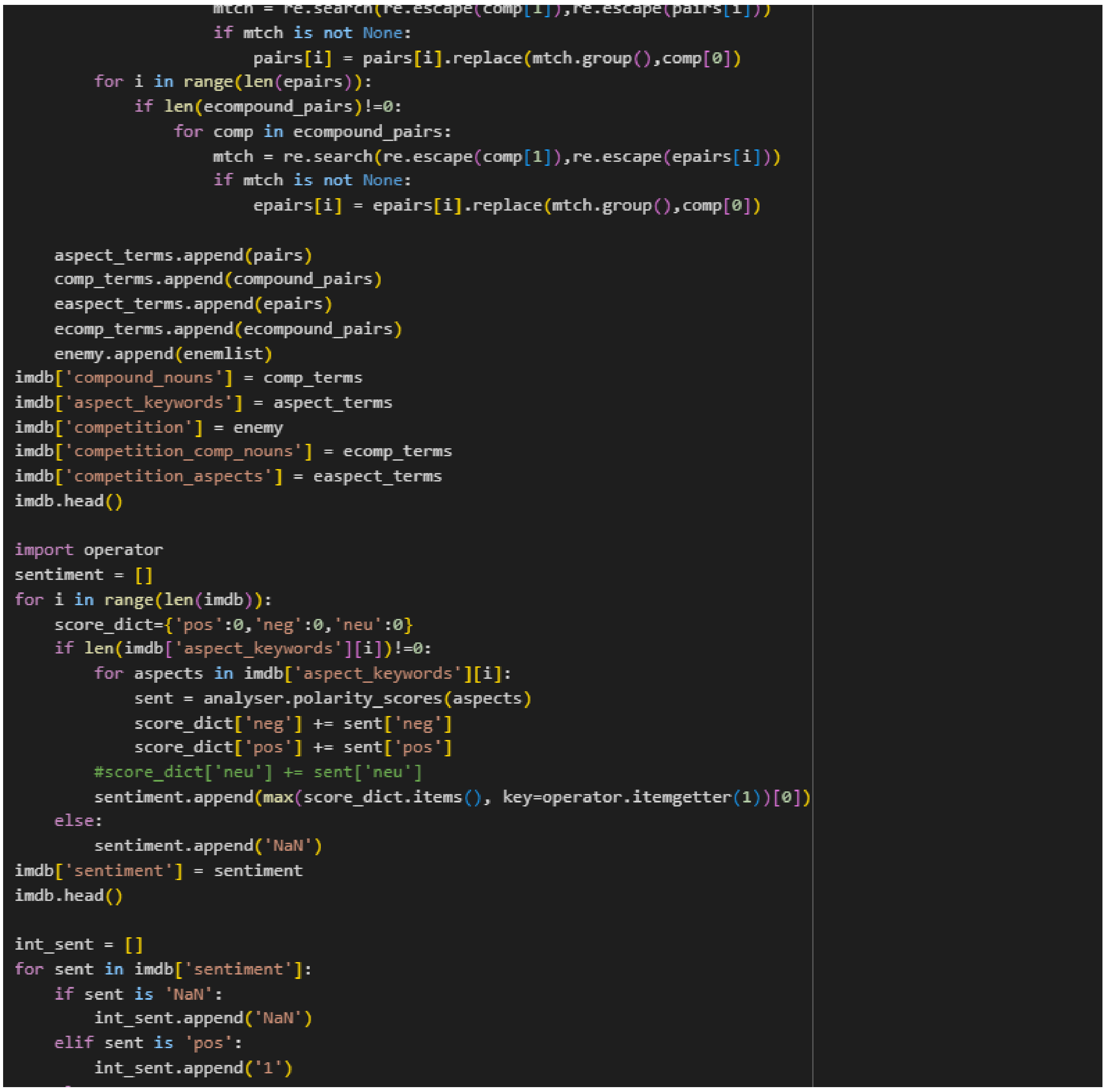
```python
from inspect import ismemberdescriptor
import numpy as np # linear algebra
import pandas as pd # data processing
import os
import re

import spacy

# Load the spaCy English language model
nlp = spacy.load("en_core_web_sm")

# reviews = imdb.get('Review', [])
reviews=imdb['Review']
competitors = []

aspect_terms = []
comp_terms = []
easpect_terms = []
ecomp_terms = []
enemy = []
for x in tqdm(range(len(reviews))):
    review_text = reviews[x]
    amod_pairs = []
    advmod_pairs = []
    compound_pairs = []
    xcomp_pairs = []
    neg_pairs = []
    eamod_pairs = []
    eadvmod_pairs = []
    ecompound_pairs = []
    eneg_pairs = []
    excomp_pairs = []
    enemlist = []
    if len(str(imdb['Review'][x])) != 0:
        lines = str(imdb['Review'][x]).replace('*',' ').replace('-',' ').replace('so ',' ').replace('be ',' ').replace('are ',' ').replace('just ',' ').replace('get ','').replace('were ',' ').replace('When ','').replace('when ','').replace('again ','
        for line in lines:
            enem_list = []
            for eny in competitors:
                enem = re.search(eny,line)
                if enem is not None:
                    enem_list.append(enem.group())
            if len(enem_list)==0:
                doc = nlp(line)
                str1=''
```



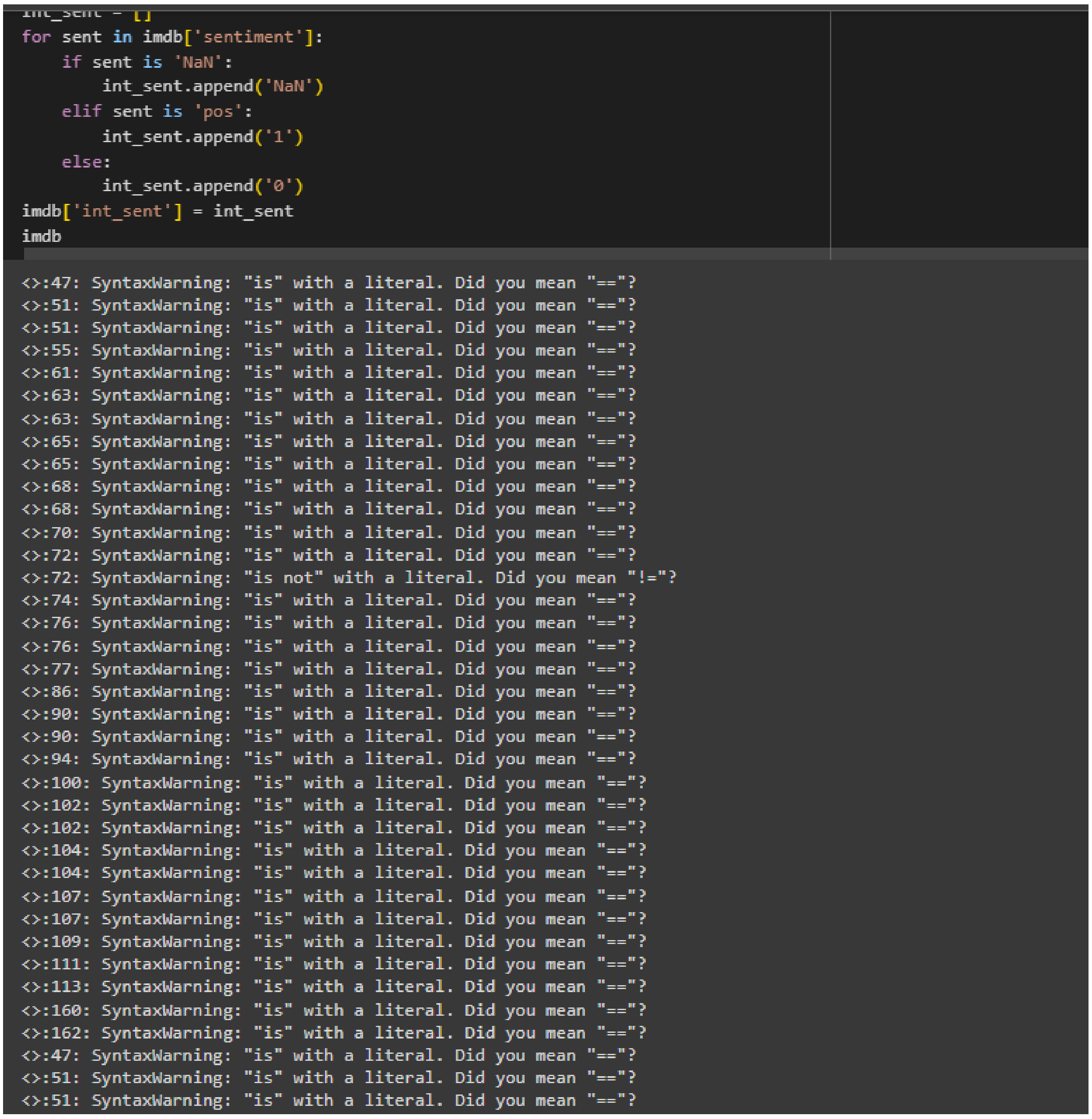
```python
                    enem_list.append(enem.group())
            if len(enem_list)==0:
                doc = nlp(line)
                str1=''
                str2=''
                for token in doc:
                    if token.pos_ is 'NOUN':
                        for j in token.lefts:
                            if j.dep_ == 'compound':
                                compound_pairs.append((j.text+' '+token.text,token.text))
                            if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary condition
                                str1 = j.text+' '+token.text
                                amod_pairs.append(j.text+' '+token.text)
                                for k in j.lefts:
                                    if k.dep_ is 'advmod': #secondary condition to get adjective of adjectives
                                        str2 = k.text+' '+j.text+' '+token.text
                                        amod_pairs.append(k.text+' '+j.text+' '+token.text)
                                mtch = re.search(re.escape(str1),re.escape(str2))
                                if mtch is not None:
                                    amod_pairs.remove(str1)
                    if token.pos_ is 'VERB':
                        for j in token.lefts:
                            if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                                advmod_pairs.append(j.text+' '+token.text)
                            if j.dep_ is 'neg' and j.pos_ is 'ADV':
                                neg_pairs.append(j.text+' '+token.text)
                        for j in token.rights:
                            if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                                advmod_pairs.append(token.text+' '+j.text)
                    if token.pos_ is 'ADJ':
                        for j,h in zip(token.rights,token.lefts):
                            if j.dep_ is 'xcomp' and h.dep_ is not 'neg':
                                for k in j.lefts:
                                    if k.dep_ is 'aux':
                                        xcomp_pairs.append(token.text+' '+k.text+' '+j.text)
                            elif j.dep_ is 'xcomp' and h.dep_ is 'neg':
                                if k.dep_ is 'aux':
                                        neg_pairs.append(h.text +' '+token.text+' '+k.text+' '+j.text)

            else:
                enemlist.append(enem_list)
                doc = nlp(line)
                str1=''
                str2=''
```



```python
    else:
        enemlist.append(enem_list)
        doc = nlp(line)
        str1=''
        str2=''
        for token in doc:
            if token.pos_ is 'NOUN':
                for j in token.lefts:
                    if j.dep_ == 'compound':
                        ecompound_pairs.append((j.text+' '+token.text,token.text))
                    if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary condition
                        str1 = j.text+' '+token.text
                        eamod_pairs.append(j.text+' '+token.text)
                        for k in j.lefts:
                            if k.dep_ is 'advmod': #secondary condition to get adjective of adjectives
                                str2 = k.text+' '+j.text+' '+token.text
                                eamod_pairs.append(k.text+' '+j.text+' '+token.text)
                        mtch = re.search(re.escape(str1),re.escape(str2))
                        if mtch is not None:
                            eamod_pairs.remove(str1)
            if token.pos_ is 'VERB':
                for j in token.lefts:
                    if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                        eadvmod_pairs.append(j.text+' '+token.text)
                    if j.dep_ is 'neg' and j.pos_ is 'ADV':
                        eneg_pairs.append(j.text+' '+token.text)
                for j in token.rights:
                    if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                        eadvmod_pairs.append(token.text+' '+j.text)
            if token.pos_ is 'ADJ':
                for j in token.rights:
                    if j.dep_ is 'xcomp':
                        for k in j.lefts:
                            if k.dep_ is 'aux':
                                excomp_pairs.append(token.text+' '+k.text+' '+j.text)
pairs = list(set(amod_pairs+advmod_pairs+neg_pairs+xcomp_pairs))
epairs = list(set(eamod_pairs+eadvmod_pairs+eneg_pairs+excomp_pairs))
for i in range(len(pairs)):
    if len(compound_pairs)!=0:
        for comp in compound_pairs:
            mtch = re.search(re.escape(comp[1]),re.escape(pairs[i]))
            if mtch is not None:
                pairs[i] = pairs[i].replace(mtch.group(),comp[0])
for i in range(len(epairs)):
```

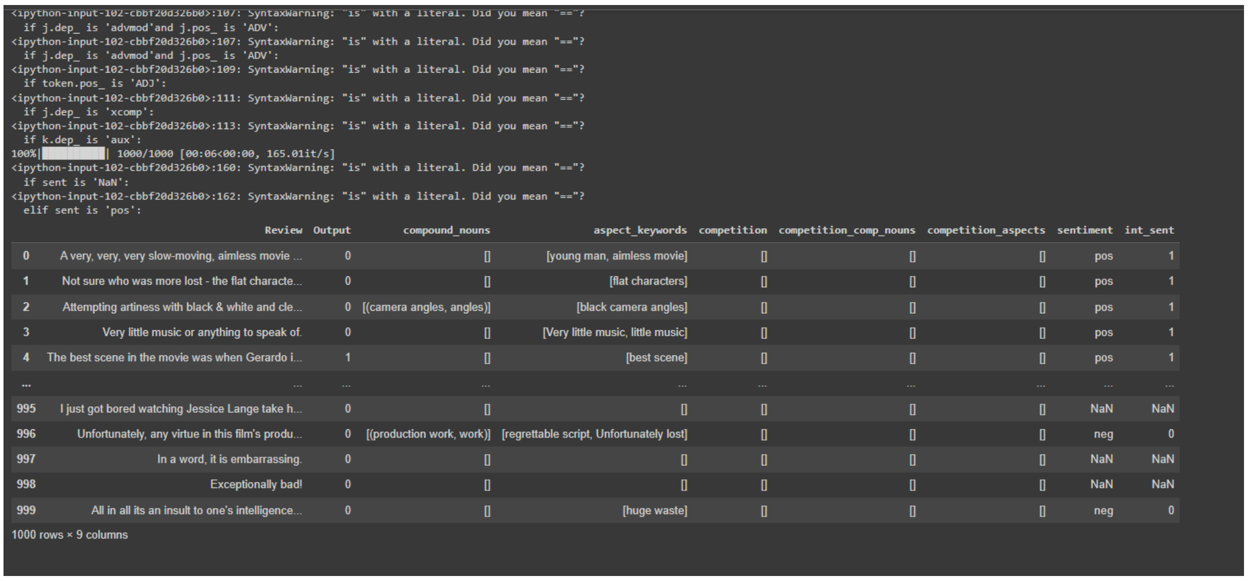


```
                    mtch = re.search(re.escape(comp[1]),re.escape(pairs[i]))
                    if mtch is not None:
                        pairs[i] = pairs[i].replace(mtch.group(),comp[0])
        for i in range(len(epairs)):
            if len(ecompound_pairs)!=0:
                for comp in ecompound_pairs:
                    mtch = re.search(re.escape(comp[1]),re.escape(epairs[i]))
                    if mtch is not None:
                        epairs[i] = epairs[i].replace(mtch.group(),comp[0])

    aspect_terms.append(pairs)
    comp_terms.append(compound_pairs)
    easpect_terms.append(epairs)
    ecomp_terms.append(ecompound_pairs)
    enemy.append(enemlist)
imdb['compound_nouns'] = comp_terms
imdb['aspect_keywords'] = aspect_terms
imdb['competition'] = enemy
imdb['competition_comp_nouns'] = ecomp_terms
imdb['competition_aspects'] = easpect_terms
imdb.head()

import operator
sentiment = []
for i in range(len(imdb)):
    score_dict={'pos':0,'neg':0,'neu':0}
    if len(imdb['aspect_keywords'][i])!=0:
        for aspects in imdb['aspect_keywords'][i]:
            sent = analyser.polarity_scores(aspects)
            score_dict['neg'] += sent['neg']
            score_dict['pos'] += sent['pos']
        #score_dict['neu'] += sent['neu']
        sentiment.append(max(score_dict.items(), key=operator.itemgetter(1))[0])
    else:
        sentiment.append('NaN')
imdb['sentiment'] = sentiment
imdb.head()

int_sent = []
for sent in imdb['sentiment']:
    if sent is 'NaN':
        int_sent.append('NaN')
    elif sent is 'pos':
        int_sent.append('1')
```



```
for sent in imdb['sentiment']:
    if sent is 'NaN':
        int_sent.append('NaN')
    elif sent is 'pos':
        int_sent.append('1')
    else:
        int_sent.append('0')
imdb['int_sent'] = int_sent
imdb
```

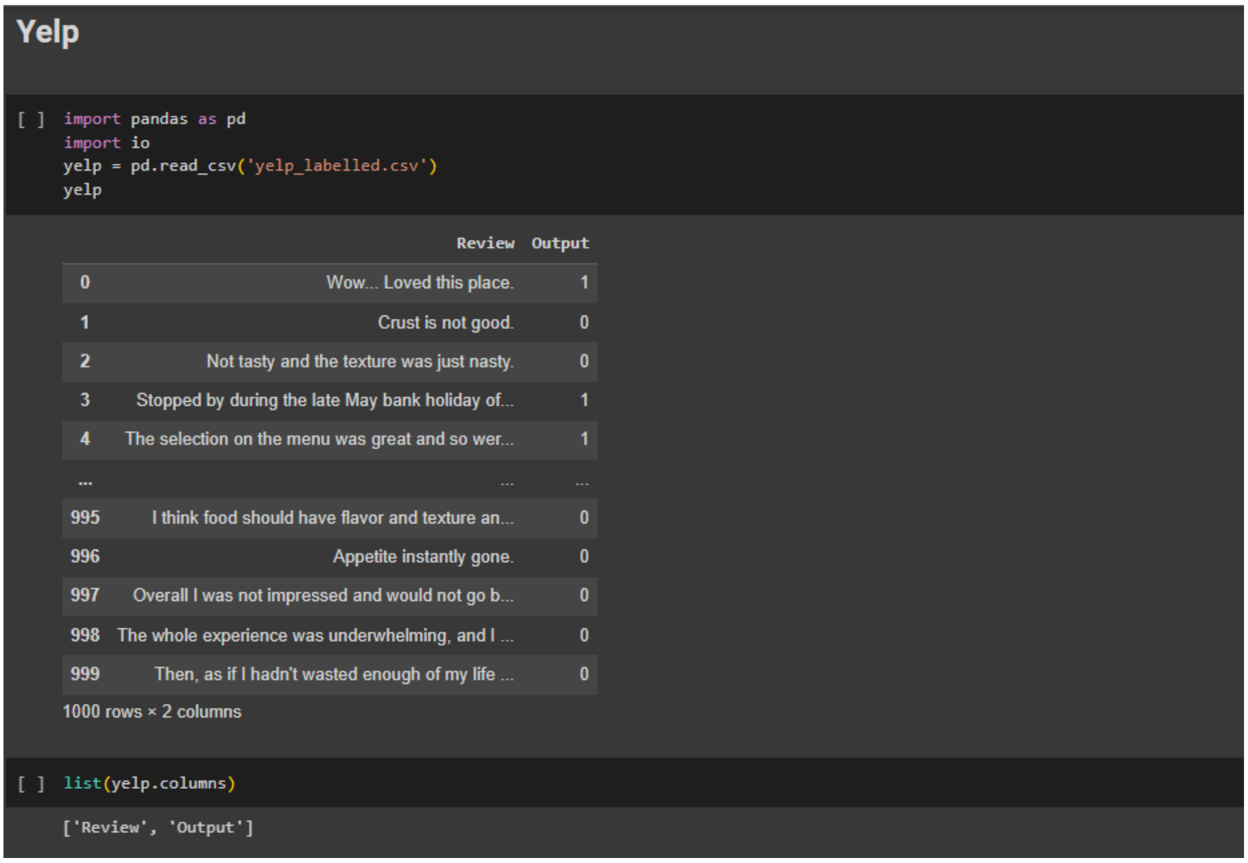
```
<>:47: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:51: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:51: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:55: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:61: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:63: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:63: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:65: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:65: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:68: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:68: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:70: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:72: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:72: SyntaxWarning: "is not" with a literal. Did you mean "!="?
<>:74: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:76: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:76: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:77: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:86: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:90: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:90: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:94: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:100: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:102: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:102: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:104: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:104: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:107: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:107: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:109: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:111: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:113: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:160: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:162: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:47: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:51: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:51: SyntaxWarning: "is" with a literal. Did you mean "=="?
```



```
<ipython-input-102-cbbf20d326b0>:107: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if j.dep_ is 'advmod'and j.pos_ is 'ADV':
<ipython-input-102-cbbf20d326b0>:107: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if j.dep_ is 'advmod'and j.pos_ is 'ADV':
<ipython-input-102-cbbf20d326b0>:109: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if token.pos_ is 'ADJ':
<ipython-input-102-cbbf20d326b0>:111: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if j.dep_ is 'xcomp':
<ipython-input-102-cbbf20d326b0>:113: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if k.dep_ is 'aux':
100%|██████████| 1000/1000 [00:06<00:00, 165.01it/s]
<ipython-input-102-cbbf20d326b0>:160: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if sent is 'NaN':
<ipython-input-102-cbbf20d326b0>:162: SyntaxWarning: "is" with a literal. Did you mean "=="?
  elif sent is 'pos':
```

	Review	Output	compound_nouns	aspect_keywords	competition	competition_comp_nouns	competition_aspects	sentiment	int_sent
0	A very, very, very slow-moving, aimless movie ...	0	[]	[young man, aimless movie]	[]	[]	[]	pos	1
1	Not sure who was more lost - the flat characte...	0	[]	[flat characters]	[]	[]	[]	pos	1
2	Attempting artiness with black & white and cle...	0	[(camera angles, angles)]	[black camera angles]	[]	[]	[]	pos	1
3	Very little music or anything to speak of.	0	[]	[Very little music, little music]	[]	[]	[]	pos	1
4	The best scene in the movie was when Gerardo i...	1	[]	[best scene]	[]	[]	[]	pos	1
...	...	...	...	...	...	...	...	...	...
995	I just got bored watching Jessice Lange take h...	0	[]	[]	[]	[]	[]	NaN	NaN
996	Unfortunately, any virtue in this film's produ...	0	[(production work, work)]	[regrettable script, Unfortunately lost]	[]	[]	[]	neg	0
997	In a word, it is embarrassing.	0	[]	[]	[]	[]	[]	NaN	NaN
998	Exceptionally bad!	0	[]	[]	[]	[]	[]	NaN	NaN
999	All in all its an insult to one's intelligence...	0	[]	[huge waste]	[]	[]	[]	neg	0

1000 rows × 9 columns



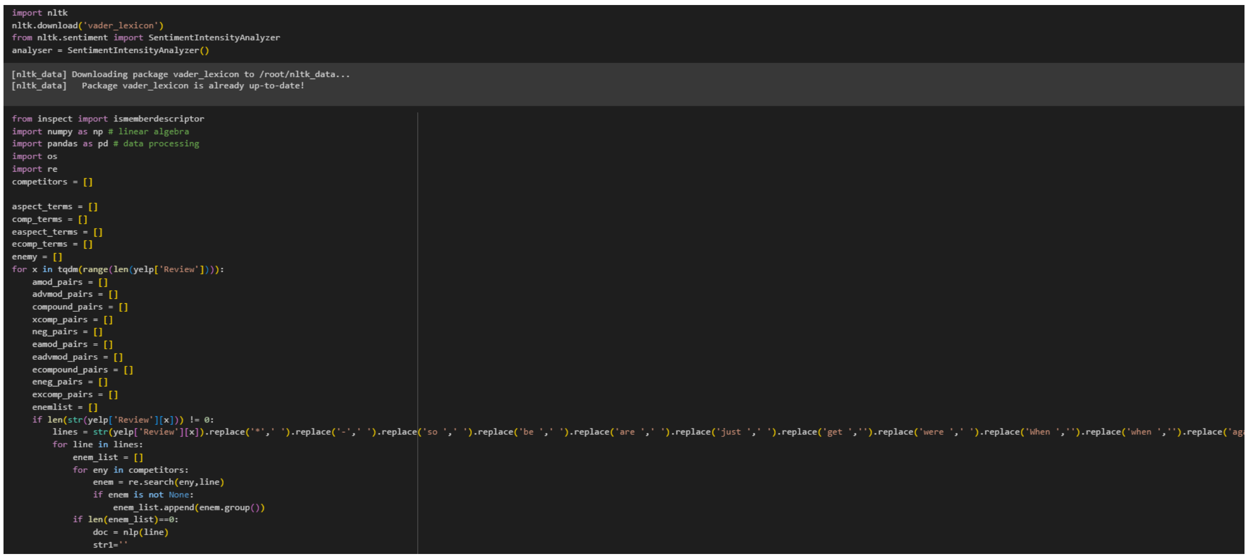
Yelp

```
[ ] import pandas as pd
    import io
    yelp = pd.read_csv('yelp_labelled.csv')
    yelp
```

	Review	Output
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1
...	...	...
995	I think food should have flavor and texture an...	0
996	Appetite instantly gone.	0
997	Overall I was not impressed and would not go b...	0
998	The whole experience was underwhelming, and I ...	0
999	Then, as if I hadn't wasted enough of my life ...	0

1000 rows × 2 columns

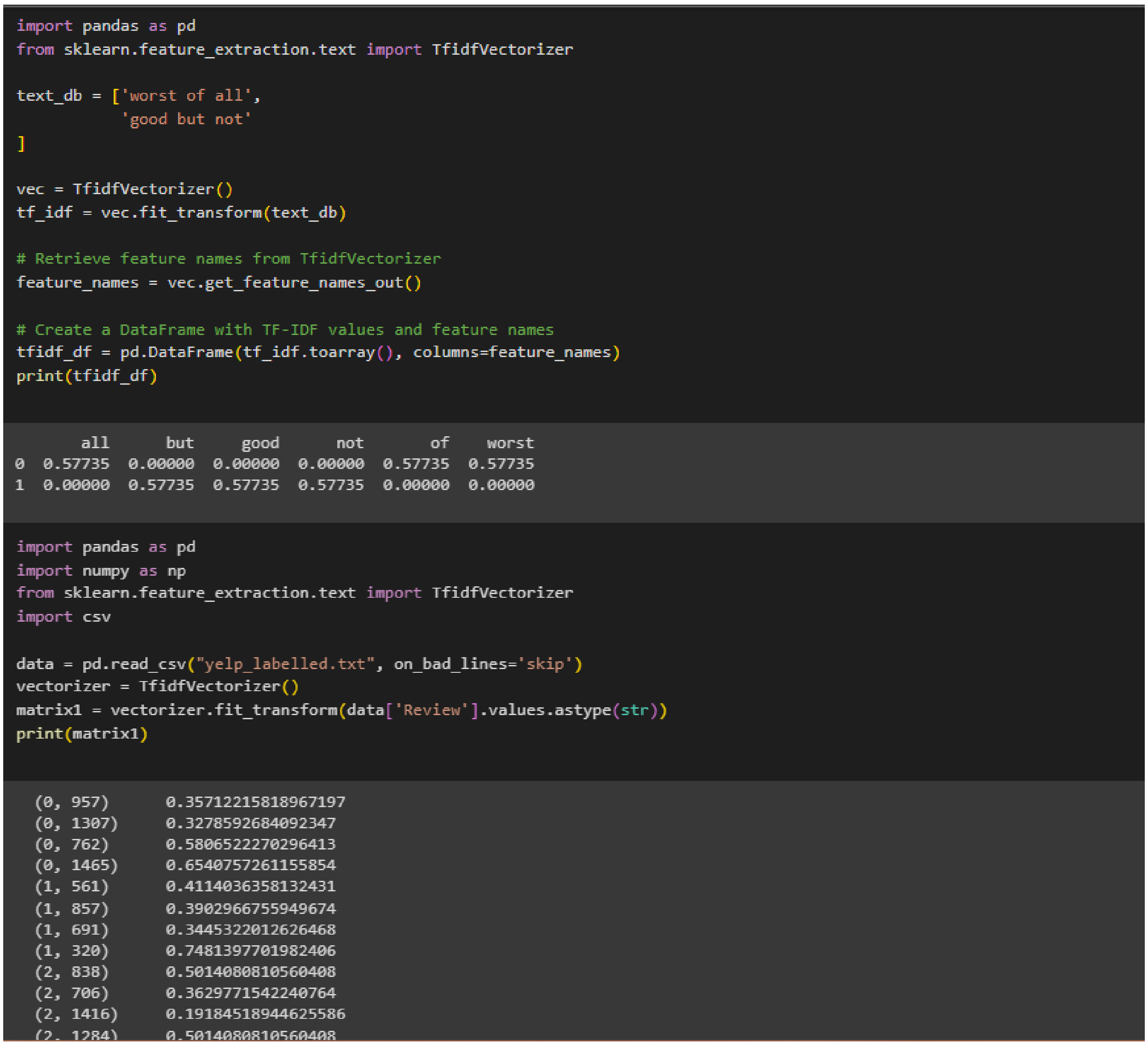
```
[ ] list(yelp.columns)

    ['Review', 'Output']
```



```
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```
from inspect import ismemberdescriptor
import numpy as np # linear algebra
import pandas as pd # data processing
import os
import re
competitors = []

aspect_terms = []
comp_terms = []
easpect_terms = []
ecomp_terms = []
enemy = []
for x in tqdm(range(len(yelp['Review']))):
    amod_pairs = []
    advmod_pairs = []
    compound_pairs = []
    xcomp_pairs = []
    neg_pairs = []
    eamod_pairs = []
    eadvmod_pairs = []
    ecompound_pairs = []
    eneg_pairs = []
    excomp_pairs = []
    enemlist = []
    if len(str(yelp['Review'][x])) != 0:
        lines = str(yelp['Review'][x]).replace('*',' ').replace('-',' ').replace('so ',' ').replace('be ',' ').replace('are ',' ').replace('just ',' ').replace('get ','').replace('were ',' ').replace('When ','').replace('when ','').replace('ag
        for line in lines:
            enem_list = []
            for eny in competitors:
                enem = re.search(eny,line)
                if enem is not None:
                    enem_list.append(enem.group())
            if len(enem_list)==0:
                doc = nlp(line)
                str1=''
```

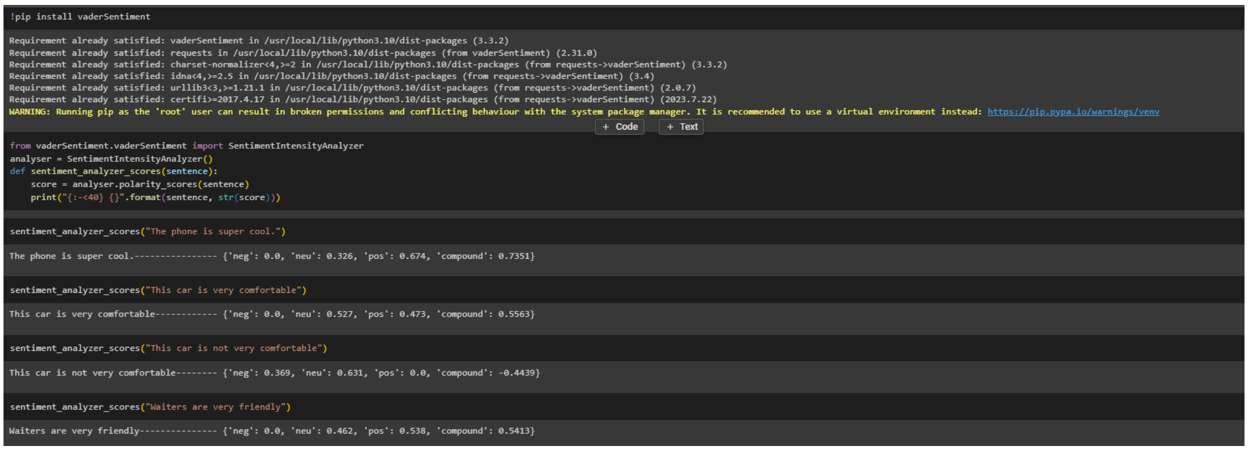


```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

text_db = ['worst of all',
           'good but not'
]

vec = TfidfVectorizer()
tf_idf = vec.fit_transform(text_db)

# Retrieve feature names from TfidfVectorizer
feature_names = vec.get_feature_names_out()

# Create a DataFrame with TF-IDF values and feature names
tfidf_df = pd.DataFrame(tf_idf.toarray(), columns=feature_names)
print(tfidf_df)
```

```
       all      but     good      not       of    worst
0  0.57735  0.00000  0.00000  0.00000  0.57735  0.57735
1  0.00000  0.57735  0.57735  0.57735  0.00000  0.00000
```

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
import csv

data = pd.read_csv("yelp_labelled.txt", on_bad_lines='skip')
vectorizer = TfidfVectorizer()
matrix1 = vectorizer.fit_transform(data['Review'].values.astype(str))
print(matrix1)
```

```
  (0, 957)      0.35712215818967197
  (0, 1307)     0.3278592684092347
  (0, 762)      0.5806522270296413
  (0, 1465)     0.6540757261155854
  (1, 561)      0.4114036358132431
  (1, 857)      0.39029667755949674
  (1, 691)      0.3445322012626468
  (1, 320)      0.7481397701982406
  (2, 838)      0.5014080810560408
  (2, 706)      0.3629771542240764
  (2, 1416)     0.19184518944625586
  (2, 1284)     0.5014080810560408
```



```
!pip install vaderSentiment

Requirement already satisfied: vaderSentiment in /usr/local/lib/python3.10/dist-packages (3.3.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from vaderSentiment) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2023.7.22)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

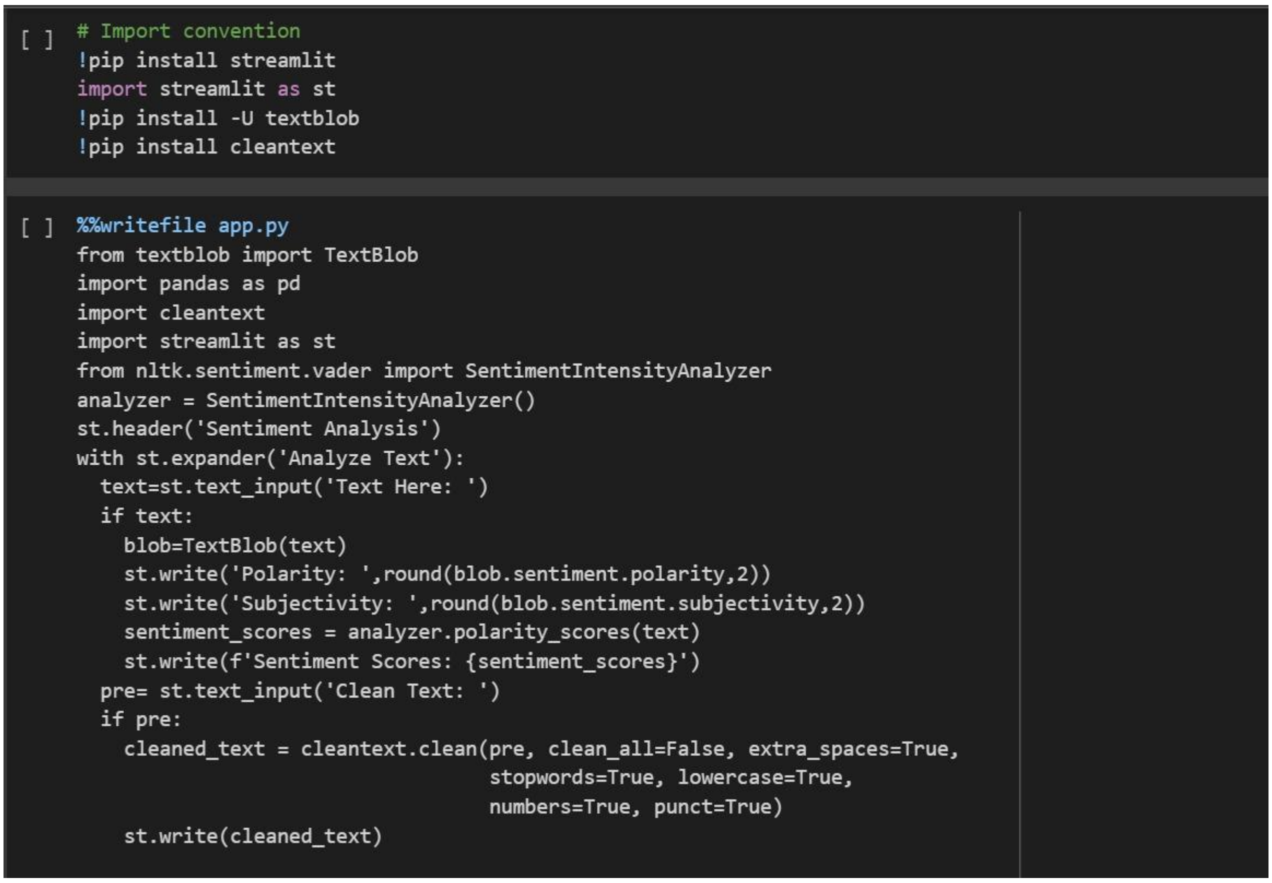
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()
def sentiment_analyzer_scores(sentence):
    score = analyser.polarity_scores(sentence)
    print("{:-<40} {}".format(sentence, str(score)))
```

```
sentiment_analyzer_scores("The phone is super cool.")

The phone is super cool.---------------- {'neg': 0.0, 'neu': 0.326, 'pos': 0.674, 'compound': 0.7351}
```

```
sentiment_analyzer_scores("This car is very comfortable")

This car is very comfortable------------ {'neg': 0.0, 'neu': 0.527, 'pos': 0.473, 'compound': 0.5563}
```

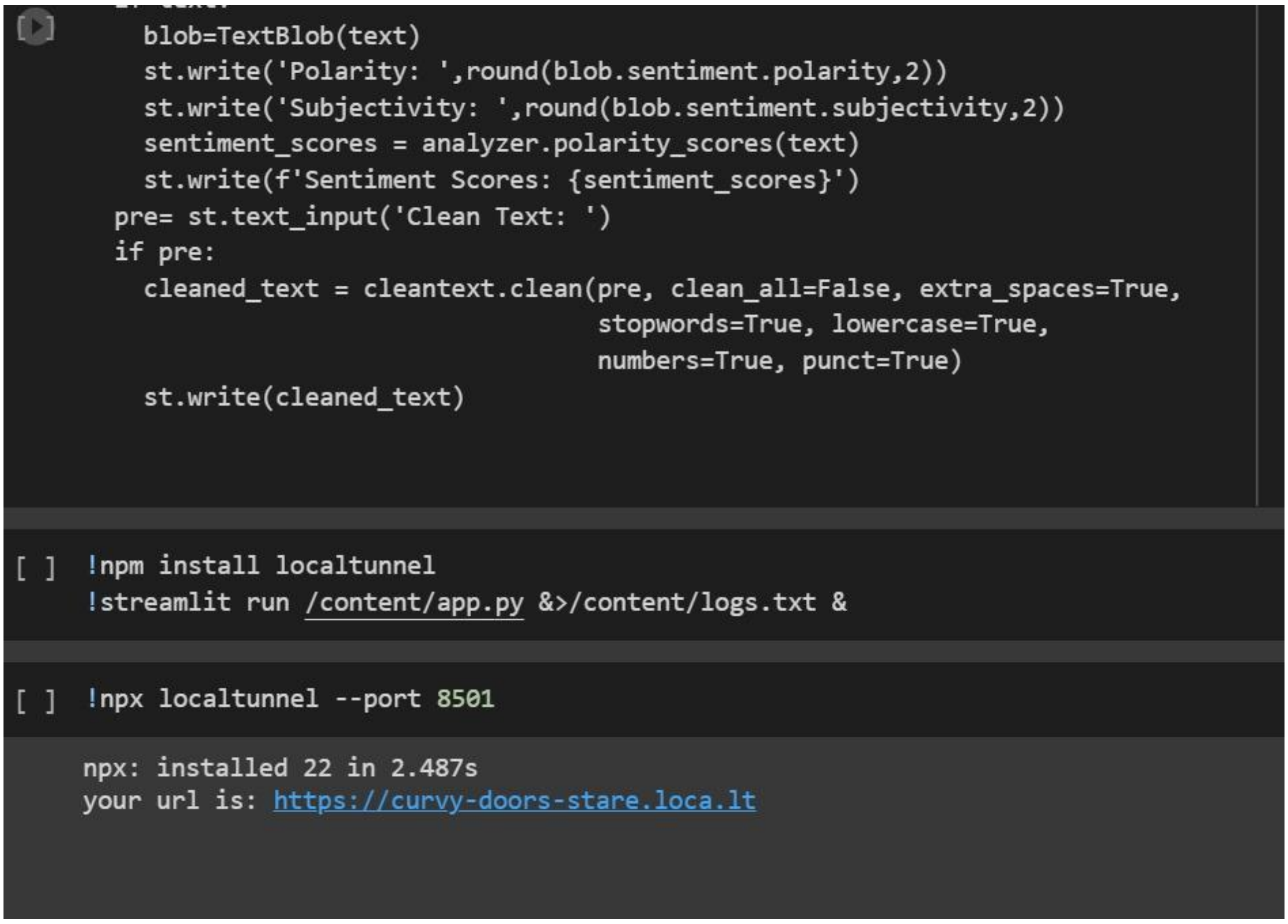
```
sentiment_analyzer_scores("This car is not very comfortable")

This car is not very comfortable-------- {'neg': 0.369, 'neu': 0.631, 'pos': 0.0, 'compound': -0.4439}
```

```
sentiment_analyzer_scores("Waiters are very friendly")

Waiters are very friendly--------------- {'neg': 0.0, 'neu': 0.462, 'pos': 0.538, 'compound': 0.5413}
```



```
[ ]  # Import convention
     !pip install streamlit
     import streamlit as st
     !pip install -U textblob
     !pip install cleantext
```

```
[ ]  %%writefile app.py
     from textblob import TextBlob
     import pandas as pd
     import cleantext
     import streamlit as st
     from nltk.sentiment.vader import SentimentIntensityAnalyzer
     analyzer = SentimentIntensityAnalyzer()
     st.header('Sentiment Analysis')
     with st.expander('Analyze Text'):
       text=st.text_input('Text Here: ')
       if text:
         blob=TextBlob(text)
         st.write('Polarity: ',round(blob.sentiment.polarity,2))
         st.write('Subjectivity: ',round(blob.sentiment.subjectivity,2))
         sentiment_scores = analyzer.polarity_scores(text)
         st.write(f'Sentiment Scores: {sentiment_scores}')
       pre= st.text_input('Clean Text: ')
       if pre:
         cleaned_text = cleantext.clean(pre, clean_all=False, extra_spaces=True,
                                        stopwords=True, lowercase=True,
                                        numbers=True, punct=True)
         st.write(cleaned_text)
```



```
    blob=TextBlob(text)
    st.write('Polarity: ',round(blob.sentiment.polarity,2))
    st.write('Subjectivity: ',round(blob.sentiment.subjectivity,2))
    sentiment_scores = analyzer.polarity_scores(text)
    st.write(f'Sentiment Scores: {sentiment_scores}')
  pre= st.text_input('Clean Text: ')
  if pre:
    cleaned_text = cleantext.clean(pre, clean_all=False, extra_spaces=True,
                                   stopwords=True, lowercase=True,
                                   numbers=True, punct=True)
    st.write(cleaned_text)
```

```
[ ] !npm install localtunnel
    !streamlit run /content/app.py &>/content/logs.txt &
```

```
[ ] !npx localtunnel --port 8501
```

```
npx: installed 22 in 2.487s
your url is: https://curvy-doors-stare.loca.lt
```

REFERENCES

1. A. García-Pablos *et al. An almost*

Based Sentiment Analysis

Expert Systems with Applications (2018)

2. M. Fernández-Gavilanes *et al.* Unsupervised method for sentiment analysis in online texts

Expert Systems with Applications (2016)

3. M.H. Alam *et al.* Joint multi-grain topic sentiment: Modeling semantic aspects for online reviews Information Sciences (2016)

4. M.S. Akhtar *et al.* Feature selection and ensemble construction: A two-step *aspect-based* Knowledge-Based Systems (2017)

5. M.S. Akhtar *et al.* Aspect-based sentiment analysis in Hindi: Resource creation and evaluation

6. M. References, X. Apidianaki, C. Tannier, and . Richart, A Dataset for Aspect-Based Sentiment Analysis in French, Proceedings of the International Conference on Language Resources and Evaluation, 2016.

7. E. Cambria, W. Björn, Y. Schuller, C. Xia, and . Havasi, New Avenues in Opinion Mining and Sentiment Analysis, IEEE Intelligent Systems, vol.28, issue.2, pp.15-21, 2013.

8. O. De-clercq and V. Hoste, Rude waiters but mouthwatering pastries! An exploratory study into Dutch Aspect-Based Sentiment Analysis, Proceedings of the 10th International Conference on Language Resources and Evaluation, 2016.