ASSESSMENT COVER SHEET

Assignment Details

Course: Applied Natural Language Processing________________

Semester/Academic Year: Sem 1 2023________________

Assignment title: Building an aspect-based sentiment analysis algorithm based on syntactic parsing___________

Assessment Criteria

Assessment Criteria are included in the Assignment Descriptions that are published on each course's website.

Plagiarism and Collusion

Plagiarism: using another person's ideas, designs, words, or works without appropriate acknowledgment.

Collusion: another person assisting in the production of an assessment submission without the express requirement, consent or knowledge of the assessor.

Consequences of Plagiarism and Collusion

The penalties associated with plagiarism and collusion are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity. Penalties may include the requirement to revise and resubmit assessment work, receiving a result of zero for the assessment work, failing the course, expulsion, and/or receiving a financial penalty.

declaration

I declare that all material in this assessment is my own work except where there is clear acknowledgment and reference to the work of others. I have read the University Policy Statement on Plagiarism, Collusion, and Related Forms of Cheating:

mttp://www.adelaide.edu.au/policies/?230

I give permission for my assessment work to be reproduced and submitted to academic staff for the purposes of assessment and to be copied, submitted, and retained in a form suitable for electronic checking of plagiarism.

Yash Kasundra (A1838670)
23/06/2023

SIGNATURE AND DATE

Introduction

When examining sentiments towards particular aspects or aspect phrases, aspect-based sentiment analysis, or ABSA, is essential. The purpose of this project is to create a robust ABSA system using syntactic parsing methods. We want to create a piece of code that can accurately anticipate the sentiment polarities (positive, negative, or neutral) connected to particular aspect phrases.

Let's look at few ABSA examples to better understand the idea. Take into account the phrase, "The camera quality is excellent, but the battery life is poor." In this case, the terms "camera quality" and "battery life" each have a good and negative connotation. Another illustration is the phrase, "The food was disappointing, but the service was prompt." The terms "service" and "food" with positive and negative connotations, respectively, are the aspect terms in this instance.

The goal of this research is to create a system that properly predicts the sentiment polarities related to each aspect term using a dataset that contains aspect phrases. To do this, we'll build rules based on the findings of syntactic parsing using methods like constituency parsing, dependency parsing, or a combination of the two.

Let's use the following rule as an example: The attitude towards the aspect word is positive if the child of the aspect term is a modifier with the dependency type "amod," such as "excellent," "outstanding," or "superb." In our ABSA system, this rule illustrates how syntactic parsing can be used to calculate sentiment polarity.

The ABSA system will offer a reliable solution for sentiment analysis of aspect phrases by applying a syntactic parsing-based technique. We can effectively forecast sentiment polarities by carefully designing rules based on syntactic parsing results, enabling a deeper grasp of customer thoughts and feedback towards particular areas.

1. Dataset Loading and EDA:

A dataset of restaurant reviews was employed for this project. A collection of restaurant reviews make up the dataset. Restaurants.xml is the name of the XML file containing the data. Identifier, text, aspect term, aspect term polarity, aspect category, and aspect category polarity are present in the file. The review text is included in the text. The term for which the sentiment will be examined is the aspect term. Initially, restaurant.xml had 3041 data then I divided the data into training and testing sets, the training set contains 2337 reviews, while the testing set contains 1002. After removing all the conflicted sentiment data from both training and testing we had 2273 reviews for training and 979 for testing. A term's polarity or attitude can have one of three possible values: positive-negative, or neutral. Below Images shows the histogram plot on lables for both training and testing set after preprocessing.

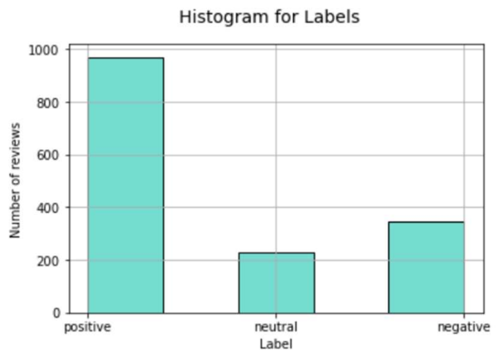


Fig-1: Histogram for training data

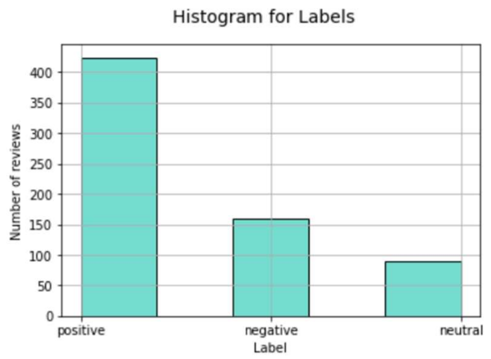


Fig-2: Histogram for testing data

Syntactic Dependency Parsing Method

A technique used in natural language processing (NLP) to examine the grammatical structure of phrases is syntactic dependency parsing. It focuses on identifying the directed edges or dependencies between words—syntactic links between words in a sentence.

Each word in a phrase is treated as a node in dependency parsing, and the connections between the nodes are shown as labeled arcs or edges. Indicating linguistic relationships like subject, object, modifier, or conjunction, these edges are typical. The final structure is frequently represented as a dependency tree. Just as shown in the below image.

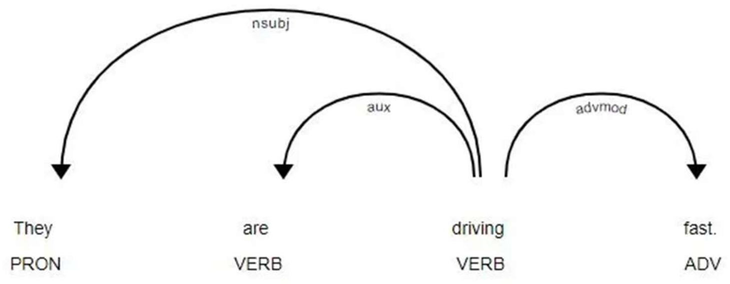


Fig-3: Syntactic Dependency Parsing Method

Dependency parsing can offer insightful information about sentence grammar that is useful for a variety of NLP tasks, such as aspect-based sentiment analysis. We can create rules and patterns to infer sentiment polarities based on the syntactic structure of the sentence by looking at the relationships between aspect terms and the words that surround them.

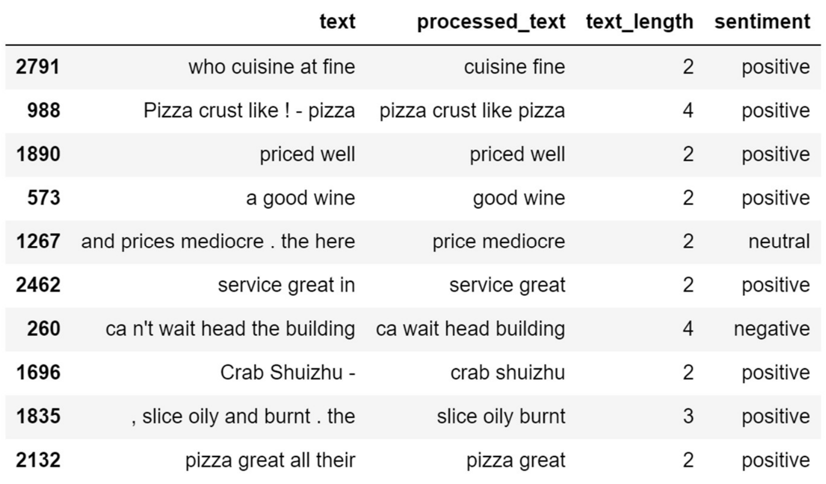
2. Preprocessing

In natural language processing (NLP), specifically for syntactic parsing, preprocessing is crucial for improving the effectiveness and quality of our sentiment analysis. We spent much time and effort on this project methodically preparing (cleaning and transforming) our textual data. These preliminary steps have been painstakingly created to improve the accuracy and efficiency of our system. Let's now explore the specific preparation techniques used on our dataset:

- **Stop Word Removal:** We recognised stopwords in our research, which are frequently used words in a language like "and," "the," and "is." These stopwords, however, frequently have minimal semantic meaning and can be safely ignored. Because of this, we eliminate stopwords from our text data during the preprocessing stage. Through this process, we can lessen noise and emphasize the significance of the words that remain, improving the accuracy of the sentiment analysis model. We can improve the accuracy and potency of the matching capabilities of our system by removing these less descriptive phrases [3].
- **Elimination of Special Characters and Punctuation:** We recognize the use of punctuation in our data, including commas, periods, question marks, and exclamation points. Although these symbols have grammatical functions, they frequently have no bearing on our study goals. Therefore, we remove punctuation from the substance of our questions throughout the preparation stage. This technique makes the text easier

to read and makes sure that our system concentrates on the semantic information and content that are most important rather than extraneous symbols. Punctuation helps to streamline the text and frees up our system to focus on the key components of the data, which enhances the effectiveness of our research as a whole[3].

- **Lemmatization:** Lemmatization is the process of changing words into lemmas, which are their lexical or root forms. Through this method, we are able to combine diverse word forms, such as plurals, verb conjugations, and several tenses, into a single representative form. By aligning similar words, lemmatization reduces lexical variation, increases text coherence, and improves matching precision. We simplify text analysis by combining word variations into their lemma counterparts, enabling improved comprehension and more precise matching results [4].
- **Drop duplicates and text with only one word:** Any duplicate rows in the text column were removed after lemmatization for both the training and testing sets of data. Additionally, single-word sentences that did not convey the sentiment of an aspect were eliminated.



	text	processed_text	text_length	sentiment
2791	who cuisine at fine	cuisine fine	2	positive
988	Pizza crust like ! - pizza	pizza crust like pizza	4	positive
1890	priced well	priced well	2	positive
573	a good wine	good wine	2	positive
1267	and prices mediocre . the here	price mediocre	2	neutral
2462	service great in	service great	2	positive
260	ca n't wait head the building	ca wait head building	4	negative
1696	Crab Shuizhu -	crab shuizhu	2	positive
1835	, slice oily and burnt . the	slice oily burnt	3	positive
2132	pizza great all their	pizza great	2	positive

Table-1: Preprocessed text

3. Model

In this project we used TF-IDF first to convert our data into vectors.

- **TF-IDF (Term Frequency-Inverse Document Frequency):**

  The TF-IDF is a popular numerical statistic that evaluates the importance of a phrase in a document inside of a collection or corpus. Inverse document frequency (IDF) and term frequency (TF) make up its two components [1].

  TF determines how often a term appears in a specific document. It provides more weight to terms that are used more frequently in the document, assuming that they are more important. To completely convey a term's relevance over the entire corpus, TF might not be able to [1].

  In contrast, IDF determines a term's rarity throughout the entire corpus. It provides terms with fewer occurrences in the corpus greater weights because it can be inferred that they have more potential for discrimination. IDF is calculated using the logarithm of the inverse fraction of documents containing the phrase[2].

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

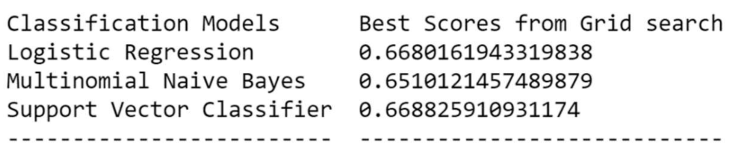
$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

Fig-4: Formula for TF-IDF [1]

- **General Classification Models:**

  In this project, we employed a diverse range of models to analyze the data. Initially, we utilized Logistic Regression, Multinomial Naïve Bayes, and Support Vector Classifier to model the data transformed by TF-IDF. Hyperparameter tuning was performed using grid search to optimize the performance of these models. To ensure reliable evaluation, cross-validation sets were created using Stratified K Fold, which maintains consistent proportions of observations with each label in every fold.

  The models were fitted with different sets of parameters, and the best-resulting accuracy on the cross-validated sets is presented in Table 2. This table provides a comprehensive overview of the performance achieved by each model's best configuration, enabling a comparative analysis of their effectiveness in handling the given dataset.



```
-------------------------  ----------------------------
Classification Models      Best Scores from Grid search
Logistic Regression        0.6680161943319838
Multinomial Naive Bayes    0.6510121457489879
Support Vector Classifier  0.668825910931174
-------------------------  ----------------------------
```

Table-2: Mean accuracy of all classification models

SVC and Logistic regression model had almost similar accuracy, but the loss was lower in logistic regression case. Thus, we used logistic regression to predict the sentiments on testing data and the confusion matrix is shown in figure-5.

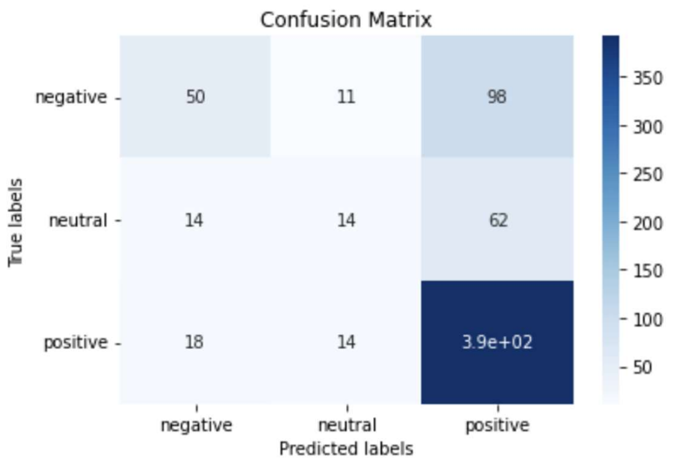


Fig-5: Confusion matrix for logistic regression on test dataset

Upon evaluating the performance of the aforementioned classification models, it was observed that logistic regression exhibited an accuracy of 67% on the test data. While this performance is commendable, it falls short of attaining optimal results. In order to enhance the model's performance, it becomes imperative to explore the realm of neural networks.

- **Neural networks**

To this end, Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) models will be leveraged, with the aim of uncovering their potential for achieving superior performance. By employing these advanced neural network architectures, we anticipate gaining deeper insights into the data and unlocking their capacity to deliver enhanced predictive accuracy. Consequently, a comprehensive comparison will be conducted to assess the relative performances of CNN and LSTM models, enabling informed decision-making regarding the most suitable approach for our specific task.

We used two different neural network topologies to predict sentiment. First, a neural network made up of two convolutional layers and a dense layer was built. A Bidirectional LSTM neural network was added as well. Pretrained embeddings were used to convert the processed text into vectors.

The GloVe 100-dimensional word embedding served as the pretrained embedding for this project. A huge collection of 400,000 words and their related vector representations are included in this embedding. GloVe, an unsupervised learning technique, makes it easier to map words in a space that is motivated by semantic similarity. Remember that the GloVe embedding used in this project is freely accessible and may be acquired from "http://nlp.stanford.edu/data/glove.6B.zip".

<u>Convolutional Neural Network (CNN)</u>

Convolutional neural networks (CNNs) are particularly good at modeling picture data because they are excellent at capturing local correlations within data. However, jobs involving text classification can also be accomplished with CNNs. Convolution in the context of CNNs is a linear operation that combines the input's dot product with a number of weight filters. This research developed a CNN architecture, starting with an embedding layer as the primary element.

A dropout layer with a dropout rate of 0.4 was added after the embedding layer to reduce overfitting. The Rectified Linear Unit (ReLU) was then used as the activation function for a convolutional layer with 32 filters and a stride of 3. A layer called one-dimensional max-pooling was put after this one. The next layer was a convolutional one with 64 filters and a stride of 3, which was then coupled to a global max pooling layer.

A dense layer with six filters was the next layer in the architecture, and it was followed by a second dropout layer with a dropout rate of 0.4. Three channels made up the final output layer, which used the softmax activation function. Categorical cross-entropy loss and the Adam optimizer were used to build the model. 100 training epochs were used, and the early stopping criteria monitored the validity of the results. The training batch size was adjusted to 20 to promote effective model convergence and optimization.

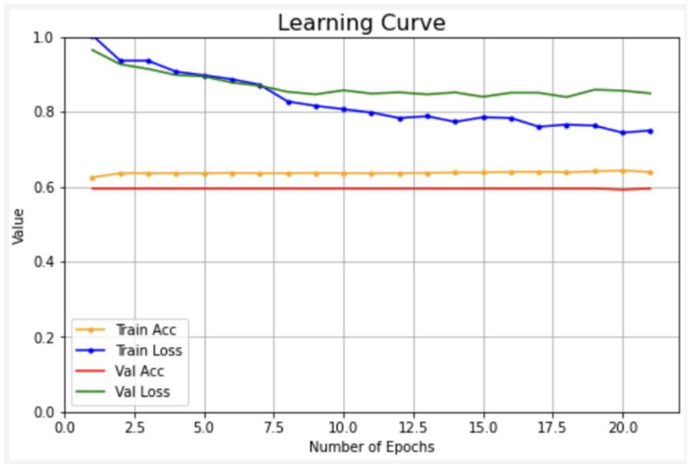
<u>**Bidirectional Long-Term Short-Term Memory Network (LSTM)**</u>

Long-term dependencies in sequential data can be modeled using LSTM (Long Short-Term Memory) layers, which are made up of interconnected memory blocks. An improved version of LSTM is known as bidirectional LSTM. Traditional recurrent neural networks' (RNNs') reliance solely on prior context is a drawback. Bidirectional RNNs use different hidden layers that are then input into the same output layer to process data in both forwards and backward directions to get around this restriction. Bidirectional LSTM, which successfully captures the long-range context in both input directions, is created by combining bidirectional RNNs and LSTMs [2].

The first layer in the LSTM network architecture is an embedding layer. A bidirectional LSTM layer with 20 units and a dropout rate of 0.5 follows this. Following the use of a dense layer with 10 filters and the ReLU activation function, a dense layer with 3 output channels and softmax activation is created. Categorical cross-entropy loss and the RMSprop optimizer are used to build the model.

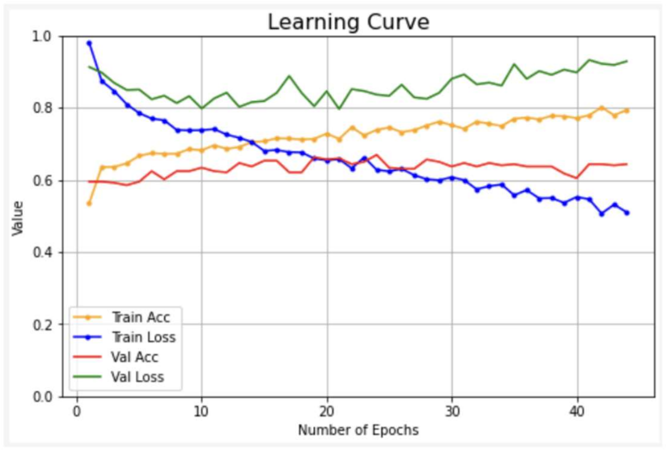
100 training epochs are used, and the early stopping criteria monitor the validity of the results. The training batch size is set at 20 to allow for effective model optimization and convergence.

From the training set we created validation set which was 20% of the current training data and based on that we evaluated our neural network models. Below graph shows the different learning curves for both models.



```
Validation set
loss: 0.9658093452453613
accuracy: 0.5954692363739014
```

Fig-6: Learning Curves for CNN model



```
Validation set
loss: 0.8369725346565247
accuracy: 0.6699029207229614
```

Fig-6: Learning Curves for LSTM model

The CNN model indicates a situation of high bias and low variance thanks to the declining training loss and validation loss, as well as the comparable training accuracy and validation accuracy. This shows the model's poor data fitting and hence low accuracy in the model. While the training accuracy and validation accuracy both show a steady increase in the case of LSTM, both the training loss and validation loss continue to decline. However, in this particular case, the observed improvement in accuracy over the logistic regression model is not very significant. It is crucial to remember that doing hyperparameter adjustments may result in much-improved accuracy. Unfortunately, resource limitations prevent a thorough grid search from determining the ideal parameters for LSTM model.

TextBlob:

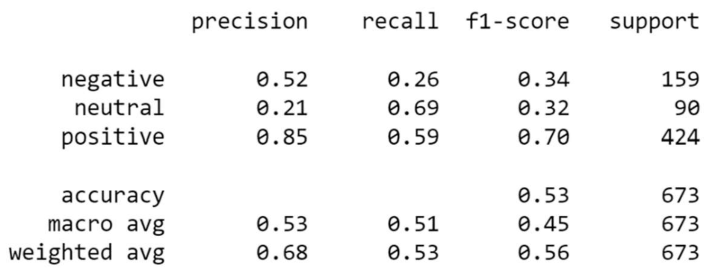
A well-known Python module for natural language processing (NLP) applications is TextBlob. For numerous NLP features, including part-of-speech tagging, noun phrase extraction, sentiment analysis, translation, and more, it offers a clear and straightforward API.

The power of TextBlob to perform sentiment analysis is one of its primary characteristics. It enables you to determine whether a sentence is positive, negative, or neutral by analyzing its sentiment or polarity. The machine learning model that underlies TextBlob's sentiment analysis has already been trained using a sizable corpus of labeled data.

Simply feed the text you wish to analyse to the TextBlob object, and then access the sentiment property, to use TextBlob's sentiment analysis functionality. The polarity and subjectivity ratings of the text are contained in a named tuple that is returned by this property. While the subjectivity score assesses subjectivity (ranging from 0 to 1, with 0 being objective and 1 being subjective), the polarity score indicates sentiment (ranging from -1 for negative to 1 for positive).

Other NLP operations including noun phrase extraction (noun_phrases), part-of-speech tagging (tags), and language translation (translate) are also supported by TextBlob.

Textblob model had f1 score of 70% on positive lables while it was really low on neutral and negative as shown in table-3.



```
              precision    recall  f1-score   support

    negative       0.52      0.26      0.34       159
     neutral       0.21      0.69      0.32        90
    positive       0.85      0.59      0.70       424

    accuracy                           0.53       673
   macro avg       0.53      0.51      0.45       673
weighted avg       0.68      0.53      0.56       673
```

Table-3: Classification report for TextBlob model

4. Conclusion:

We developed an aspect-based sentiment analysis algorithm employing machine learning models and dependency parsing. The model was evaluated primarily using the accuracy score. The Bidirectional LSTM model performed with a maximum accuracy score of 66.99% on the test set. The performance of the model has room for improvement. The neutral and negative classes cannot be reliably classified using the final model. Correctly predicting these classes may benefit from increased training data and subsequent model optimization. To enhance test performance, experiments can be conducted with different neural network models.

References:

[1] Xu, S 2018, ‘Bayesian Naïve Bayes classifiers to text classification’, Journal of Information Science, vol. 44, no. 1, pp. 48–59.

[2] Graves, A, Mohamed, A & Hinton, G 2013, ‘Speech recognition with deep recurrent neural networks’, in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, pp. 6645–6649.

[3] Saeed, M. (no date) *A guide to text preprocessing techniques for NLP - blog*, *Scale Virtual Events*. [Link : https://exchange.scale.com/public/blogs/preprocessing-techniques-in-nlp-a-guide]

[4] Saumyab (2022) Stemming vs lemmatization in NLP: Must-know differences, Analytics Vidhya. [Link: https://www.analyticsvidhya.com/blog/2022/06/stemming-vs-lemmatization-in-nlp-must-know-differences/#:~:text=Stemming%20is%20a%20process%20that,%27%20would%20return%20%27Car%27]