CS3244 Project Proposal
Group 9

Aung Naing Tun, Mavis Neo, Nicholas Jimmy Alden,
Nixon Widjaja, So Wai Yein Ethan, Sum Hung Yee

1 Introduction

1.1 Motivation

The study of the interaction between human language and computers is known as NLP or Natural Language Processing. One area in NLP that we will explore is sentiment analysis, which involves determining whether a statement is positive or negative.

Analyzing consumer sentiment is important for companies in general, as they wish to learn more about the demand for their product. Take Apple, for instance, who needs to know what its users think about the iPhone. However, with no rating system for its products, a good way to learn more about the demand for its products is by analyzing reviews made by users of the product. However, manually reading through reviews on various platforms such as social media would be time-consuming and costly, and thus these processes should be automated. By processing large amounts of data and classifying comments, Apple can gain more information about its position in the market and adapt more readily to changes in the market. Therefore, in this project, we will explore sentiment analysis, which can be beneficial for companies to understand the market and improve their products and for consumers to make informed decisions when purchasing goods. We will investigate and optimize various machine learning models, including the State-of-the-Art (SOTA) model, to perform sentiment analysis and compare their performance.

1.2 Application Outline

Amazon is one of the largest e-commerce platforms in the world, with millions of products and reviews. As a seller or a buyer, understanding the sentiment of those reviews can be critical to making informed decisions. This is where our machine learning application comes in. We aim to make a machine learning application that uses natural language processing (NLP) algorithms, i.e., sentiment analysis to analyze Amazon reviews and classify them as positive or negative based on their sentiment. This application will provide a classification summary of the overall sentiment for a product, allowing users to quickly gauge customer sentiment and make informed decisions. Our user demographics for this application are Amazon sellers, buyers, and researchers who need to analyze large volumes of Amazon reviews. From this project, we will determine from various machine learning models which one can produce the highest accuracy in classifying whether a product has a positive or negative review.

1.3 Dataset

https://www.kaggle.com/datasets/bittlingmayer/amazonreviews

Columns	Data Type	Details
Text	String	Reviews on Amazon products
Label	Integer	0: for reviews with negative sentiments 1: for reviews with positive sentiments

Since the reviews are taken straight from the Amazon site, it still contains a lot of noise from irrelevant tokens (e.g. URLs, emojis, etc.) and therefore requires cleaning, before being passed to the feature extraction process.

The label distribution seems to be pretty balanced, so there should be no need for any oversampling. From the 901,678 entries within the training set, 455,290 are labeled as positive and 446,388 are labeled as negative. The testing set contains 400,000 entries and the positive and negative labels have exactly 200,000 entries each.

For the purpose of showing the viability of our model, we will feed only the first 5,000 entries of the training set to the model and test with the first 1,000 entries of the testing set. The distribution remains more or less similar, with the training and testing set containing 2,308 and 502 positives along with 2,692 and 498 negatives, respectively.

2 Methodology

2.1 Data Preprocessing

The data is first cleaned by passing the reviews to multiple functions. Below are the several things done throughout the cleaning process:

1. ***Convert string to lowercase***
   Strings are converted to lower case to reduce the complexity of the problem. For instance, the word "Hello" and "hello" should not be treated differently.
2. ***Spell out contractions***
   Similarly, the purpose of doing this is to reduce the complexity of the problem. It is to be noted, however, that most of these resultant words will be removed in the next step.
3. ***Remove stopwords***
   Words like "actually", and "and" add little to the effect of the sentence, and thus can serve to needlessly complicate the learning process.
4. ***Remove punctuations***
   Similarly, punctuations like "." do not add any value to the text and are thus removed.
5. ***Remove URLs, numbers and special characters***
   These sequences add noise to the dataset
6. ***Lemmatize words***
   This step simplifies the problem by converting words to their base form.

We then extract features from the cleaned data to be used for our model training by using either *sklearn.feature_extraction.text.CountVectorizer (CV),* which returns a token frequency matrix, or *sklearn.feature_extraction.text.TfidfVectorizer (TV),* which returns a weighted token frequency matrix. We train the models and compare the results on two metrics:

1. If cleaning the dataset as outlined in this section results in a significant difference
2. If TV or CV is more effective

2.2 Preliminary Models

We check the viability of our dataset by passing 5,000 rows of our dataset into basic classification models (i.e. k-Nearest Neighbors and Decision Tree) and prospective improved models (i.e. Multinomial Naive Bayes and SVM)

We will compare the default model with optimized models via hyperparameter tuning. Due to the large size of our dataset, methods such as GridSearchCV will prove to be extremely inefficient. However, simply using RandomisedGridSearchCV won't provide us with an accurate enough result, since the choice is not systematic. Therefore, we will use the Bayes optimization method.

3 Results and Evaluation

Accuracy and Confusion Matrix for Tested Models

	Cleaned TV	**Cleaned CV**	**Uncleaned TV**	**Uncleaned CV**
Default kNN	***0.711*** *[[0.378 0.12] [0.169 0.333]]*	***0.653*** *[[0.283 0.215] [0.132 0.37]]*	***0.713*** *[[0.389 0.109] [0.178 0.324]]*	***0.648*** *[[0.266 0.232] [0.12 0.382]]*
Optimized kNN	***0.75*** *[[0.438 0.06] [0.19 0.312]]*	***0.681*** *[[0.305 0.193] [0.126 0.376]]*	***0.744*** *[[0.452 0.046] [0.21 0.292]]*	***0.686*** *[[0.286 0.212] [0.102 0.4]]*
Default DT	***0.705*** *[[0.344 0.154] [0.141 0.361]]*	***0.713*** *[[0.348 0.15] [0.137 0.365]]*	***0.705*** *[[0.368 0.13] [0.165 0.337]]*	***0.688*** *[[0.349 0.149] [0.163 0.339]]*
Optimized DT	***0.719*** *[[0.352 0.146] [0.135 0.367]]*	***0.712*** *[[0.355 0.143] [0.145 0.357]]*	***0.702*** *[[0.361 0.137] [0.161 0.341]]*	***0.694*** *[[0.359 0.139] [0.167 0.335]]*
Plain Bayes	***0.771*** *[[0.474 0.024] [0.205 0.297]]*	***0.813*** *[[0.434 0.064] [0.123 0.379]]*	***0.752*** *[[0.485 0.013] [0.235 0.267]]*	***0.817*** *[[0.441 0.057] [0.126 0.376]]*
Plain Linear SVM	***0.848*** *[[0.435 0.063] [0.089 0.413]]*	***0.827*** *[[0.419 0.079] [0.094 0.408]]*	***0.868*** *[[0.444 0.054] [0.078 0.424]]*	***0.815*** *[[0.414 0.084] [0.101 0.401]]*

Expectedly, KNN and DTs do not perform very well in comparison to the more advanced models, with maximum accuracy of ***0.75*** for kNN, and ***0.71*** for DT. For all of the models, cleaning the dataset generally improves the accuracy. The best performing model for this experiment is the SVM model with a linear kernel, which achieves a maximum accuracy of ***0.868***.

We observe that our dataset is viable for future work. kNN and DT already have a mean accuracy of ***0.715*** and ***0.706***. Looking at the preliminary training of plain Bayes and linear SVM (without any optimization), they achieve a mean accuracy of ***0.788*** and ***0.839***, a promising performance. We believe we can further increase the performance of our application by optimizing Bayes and SVM, along with including more sophisticated classification models, such as Recurrent Neural Networks, in particular LSTM (Long Short-Term Memory) networks.

4 Plans for Future Work

The baseline DT and kNN models do not seem to have a high enough accuracy despite our efforts at optimisation, and we might be able to improve our results with other more complex classifiers like SVM and Bayes as our improved models. However, these methods are still not optimal for sentiment analysis due to their inability to take into account the natural ordering of texts. For example, the word “not” when placed before the word “good” changes the meaning of the phrase and affects the sentiment. Therefore, we need to use a Recurrent Neural Network (RNN) model, as our state-of-the-art model.

In particular, we intend on utilizing the Bi-Directional Convolutional Long Short Term Memory (LSTM) architecture, which will consider forward and backward sequences. According to Gopalakrishnan et al., a bidirectional LSTM layer will further improve on a traditional plain LSTM model. Additionally, as proposed by the same authors, we will introduce a 1-D CNN layer to extract local features of the text, before passing these features into the subsequent bidirectional LSTM layers. In this way, our model can capture both local and global contextual information, as seen in the paper by Trueman et al.. To avoid inflating the training time, we will refrain from adding too many dense layers to the architecture.

5 Conclusion

We propose to create an ML application specializing in sentiment analysis for classifying positive and negative reviews of Amazon products, so our users can make informed decisions when purchasing goods. Our user demographics for this application are Amazon sellers, buyers, and researchers who need to analyze large volumes of Amazon reviews. We will compare various classification models and determine the one with the highest accuracy. Our preliminary results of training basic models (kNN and DT) produce reasonable accuracy, which shows that our dataset is viable for future work. We also obtain promising performance in terms of accuracy from more sophisticated models such as Naive Bayes and SVM. We plan to optimize these more complex models in the future and include Bi-Directional LSTM, which specializes in analyzing sentences from forward and backward context, as our state-of-the-art model.

6 References

1. Pedregosa et al.: *‘Scikit-learn: Machine Learning in Python’*. JMLR 12. pp. 2825-2830 (2011)
2. Harris, C.R., Millman, K.J., van der Walt, S.J. et al.: *‘Array programming with NumPy’*. Nature 585. pp. 357–362 (2020).
3. McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), Proceedings of the 9th Python in Science Conference (pp. 56-61).
4. The pandas development team. (2020, February). pandas-dev/pandas: Pandas 1.5.3. Zenodo.
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017
6. Bird, S., Loper, E., Klein, E.: *‘Natural Language Processing with Python’*. O’Reilly Media Inc (2009).
7. Gopalakrishnan, K., & Salem, F. M. (2020). Sentiment Analysis Using Simplified Long Short-term Memory Recurrent Neural Networks. CoRR, abs/2005.03993. Retrieved from https://arxiv.org/abs/2005.03993
8. J, A.K., Trueman, T.E. & Cambria, E. A Convolutional Stacked Bidirectional LSTM with a Multiplicative Attention Mechanism for Aspect Category and Sentiment Detection. Cogn Comput 13, 1423–1432 (2021)