Capstone Project – NLP Applications

Bronwyn Bowles-King

Introduction

This notebook covers the cleaning and preprocessing of a set of customer reviews, followed by sentiment analysis of these reviews using SpaCy and TextBlob. The performance of the sentiment analysis function is evaluated at the end of the notebook.

Important note

As the original dataset is very large (over 250 MB), it is in a zipped folder (zipped_dataset.zip), which needs to be upzipped before proceeding with the steps below.

0. Import packages and load the medium-sized SpaCy model

In [1]:
```
import pandas as pd
from textblob import TextBlob
import spacy
nlp = spacy.load('en_core_web_md')
```

1. Define functions

In [2]:
```
def preprocess_text(text):
    """
    Preprocess text by converting words to lowercase and lemmatising,
    removing stopwords and non-alphabetic characters.
    """

    doc = nlp(text.lower())
    tokens = [token.lemma_ for token in doc if not token.is_stop and token.is_alpha]
    return ' '.join(tokens)


def sentiment_analysis(review):
    """
    Sentiment analysis function using TextBlob for positive,
    neutral and negative sentiment detection.
    """

    analysis = TextBlob(review)

    # Get polarity score (-1 to 1)
    polarity = analysis.sentiment.polarity

    # Classify sentiment by score
    if polarity > 0.1:
        return 'Positive'
    elif polarity < -0.1:
        return 'Negative'
    else:
        return 'Neutral'
```

2. Read in the customer reviews file and prepare the text

The instructions for the task note that we can work with a sample of the dataset because of its large size, having over 30 000 rows HyperionDev (2025). I will use a subset of the first 50 rows after the table header.

This will also allow us to focus on the reviews of one specific product, which is the AmazonBasics brand of AAA batteries. The code below loads rows about this product with customer reviews included. It then checks and removes those with

blank reviews.

In [ ]:
```
# Read in only the first 50 rows after the header
file_path = "Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products_May19.csv"
data = pd.read_csv(file_path, nrows=50)

# Remove rows with missing reviews
clean_data = data.dropna(subset=['reviews.text'])

# Extract reviews.text col
reviews_data = clean_data['reviews.text']

# Apply preprocessing function to clean reviews
cleaned_reviews = reviews_data.apply(preprocess_text)

display(cleaned_reviews.head())
```

```
0    order item bad quality miss backup spring pc a...
1                      bulk expensive way product like
2                                duracell price happy
3                        work brand battery well price
4                        battery long last price great
Name: reviews.text, dtype: object
```

3. Test the sentiment analysis function on a sample of customer reviews

A random sample of 10 cleaned reviews is drawn using the pandas method sample() and sentiment analysis is run on these. A random seed is set to ensure the same sample is drawn for discussion below. To draw different random samples besides the one below, simply remove the random_state parameter.

In [4]:
```
sample_reviews_clean = cleaned_reviews.sample(n=10, random_state=81)

test_results = pd.DataFrame({
    'Review text (processed)': sample_reviews_clean,
    'Predicted sentiment': sample_reviews_clean.apply(sentiment_analysis)
})

display(test_results)
```

	Review text (processed)	Predicted sentiment
43	time buy work last long brand time charge hour...	Neutral
23	long brand name disappointed	Negative
7	look cheap non rechargeable battery perfect	Positive
38	half price	Negative
25	battery battery good price	Positive
29	light think fit light arrive nice company batt...	Positive
8	hold high power juice like energizer duracell ...	Neutral
41	satisfied purchase battery	Positive
20	second order work good brand ship door	Positive
34	use xbox controller pretty long great cheap price	Positive

4. Summary report

4.1 A description of the dataset

A list of over 30 000 consumer reviews is found in the Datafiniti Amazon Consumer Reviews file downloaded from Kaggle (2019). The data originally comes from Datafiniti's Product Database. The reviews are for products like batteries and tablets. The dataset includes detailed product information, ratings, reviews, and data likely based on how Amazon's

systems label and store information in a product database. This analysis focused on customer reviews for AmazonBasics AAA batteries within the dataset.

4.2 Details of the preprocessing steps

The preprocess_text function cleans each review in the data subset by converting all text to lowercase. It then uses spaCy's medium-sized model (md) to tokenise and lemmatise the words to reduce them to their root form, e.g. from 'ordering' to 'order'. The function also filters out stopwords (very common and short words that may not carry meaning related to sentiment), numbers and punctuation. The processed text is stored as cleaned_reviews at the end.

4.3 Evaluation of results

To see how well the model has done, I will check positive, neutral and negative labels against the original reviews from the rows displayed above. For review 41 above, the function correctly identified it as a positive response as the person used the word "satisfied". The full review is: "I am 100 satisfied with my purchase of these batteries".

However, too many words were removed from review 38 so that the program incorrectly tagged the truncated version "half price" as negative. The full review is: "These last as well as Energizer- half the price", which has two parts that indicate positive views of the product: one regarding its lifespan and the other about price. These are key aspects of customer satisfaction and the model missed both of them.

There is a different problem with the neutral tag for review 43, which reads in full: "First time I bought these they worked well and lasted almost as long as the name brands. Not the same this time around, almost no charge - 4-6 hours of run time - Junk batteries." The review is actually negative and would discourage others from buying the batteries, even if they also had a good experience with them in the past.

As the program only mathematically calculates sentiment by words in a truncated text, it cannot account for cases like this. Short but important words like "not" and "no" are being removed. Although the removal of stopwords like these speeds up the process and can help zoom in on critical information, it may be best to reconsider this for some applications and datasets where these words can be very important, such as in these customer reviews.

4.4 Insights into the model's strengths and limitations

The model's main advantage is that it is fairly easy to set-up and to get it running, making it good for beginner and intermediate programmers. However, we need to make substantial changes to the program to improve it. We can use a different, custom stopword list especially for customer- and product-related contexts. This will prevent important words and phrases customers use when giving feedback (e.g. "last as well as" in review 38) from being removed. We can also make the program more sensitive to nuanced expressions that pop up in some reviews like "half price", which shows a positive sentiment and not a statement that the product is only "half" as good as expected.

For review 41, the function missed the full extent of the person's satisfaction as the number 100 was removed, but the point here is not to classify the reviews into detailed categories. We just want to know the number of customers who are satisfied, dissatisfied or netural. So, the program has been successful in such cases where it is fairly straightforward to classify a review. One study found that TextBlob was only 56% accurate in sentiment analysis and the model built here has achieved a similar accuracy level (Shereef, 2023).

One of the weaknesses of the model is also that it takes a long time to run (although the full dataset is rather large and using a subset here was faster). SpaCy models can be computationally expensive when handling long documents or complex contents (Shereef, 2023). Methods such as training the model beforehand with labelled data and batch processing with spaCy's nlp.pipe() method can improve processing speed (StackOverflow, 2019). However, using a more advanced AI language model instead, like an enterprise-level GPT, will vastly improve the outcomes and companies today are pursuing this route to stay updated on the nuances in their customers' views (Bojić et al., 2025).

References

Bojić, L. et al. (2025). Comparing large Language models and human annotators in latent content analysis of sentiment, political leaning, emotional intensity and sarcasm. *Scientific Reports*. https://doi.org/10.1038/s41598-025-96508-3

De Camillis, P. (2022). Analysing Natural Language Processing Techniques: A Comparative Study of NLTK, spaCy, BERT, and DistilBERT on Customer Query Datasets. MSc in Data Analytics. https://arc.cct.ie/msc_da/1

HyperionDev. (2025). Capstone Project – NLP Applications Task. Prrivate repository. GitHub.

Kaggle. (2019). Consumer Reviews of Amazon Products. https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products

KGP Talkie. (2020). Amazon and IMDB Review Sentiment Classification using SpaCy. https://kgptalkie.com/amazon-and-imdb-review-sentiment-classification-using-spacy

Shereef, E. S. (2023). Sentiment Analysis in Python: TextBlob vs Vader Sentiment vs Flair vs Building It From Scratch. https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair

spaCy. (2025). English: Available trained pipelines for English. https://spacy.io/models/en

spaCy. (2025). spaCy 101: Everything you need to know. https://spacy.io/usage/spacy-101

spaCy. (2025). Vectors. https://spacy.io/api/vectors

StackOverflow. (2019). How to speed up a spacy pipeline with the nlp.pipe pattern? https://stackoverflow.com/questions/58320922/how-to-speed-up-a-spacy-pipeline-with-the-nlp-pipe-pattern