Report for problem solving with data and text A2.

What I chose to analyse

When choosing what to analyse I was really stuck on this part of the assignment. After completing part A., it took a while to find something that would be interesting to analyse but I couldn’t decide something on my own, so I selected from the prompts at the bottom of the specification document. The option I chose was the frequency distribution of reject emails vs the frequency distribution of non-reject emails. Upon gathering results for this part, I then felt it would be more beneficial to investigate the sentiment of the non-reject and reject emails. This was more interesting to analyse due to the nature of how Vader categorised reject and non-reject emails in the sentiment analysis specifically reject.

What is frequency distribution?

Frequency distribution is a way of showing how frequently (how often) values occur within a given dataset. So, in my case the resulting frequencies are how often certain values occur within the reject and non-reject data set.

What is sentiment analysis?

Sentiment analysis is when you look at a data set and categorise specific parts of the data with a feeling like positive, negative, and neutral and this allows you to make conclusions about the data.

Findings

The code which I ran to calculate the frequency analysis for rejection and non-rejection email can be found in the section marked for part b code in the document which part a is completed and, in the appendix, below.

Largely a lot of the results I got in my data show that a lot of words are shared in reject and non-reject emails. I have picked some results to analyse individually. When analysing I made the choice to remove stop words because when I tried it without there was even less variation between reject and non-reject emails so doing this made it have a bit more variation amongst emails.

After I conducted the findings for frequency distribution, I felt that I wanted to investigate further about the nature of the reject and non-reject emails, so I conducted sentiment analysis upon the reject and non-reject emails. This was beneficial because I could use a pre trained model in Vader which would categorise the emails as positive, negative, and neutral according to certain values associated with sentiment analysis. The data that came about from this was interesting because for reject emails I feel it mostly categorised what I would deem negative sentiment emails as positive or netural as they are rejecting someone from a job. However, I agree more with the conclusions for non-reject emails Vader made more detailed explanations of findings are below.

Top 3 results for frequency distribution in rejection

In rejection emails the top frequency was the word interest with a value of 0.030181086519114688(75 times) and interest does appear in non-reject, but it wasn’t one of the top results. It had a frequency of 0.0036848072562358277(13 times).

The 2nd most frequent word that appears in rejection emails is Thank which largely makes sense as these are people being rejected from job applications with a frequency value of 0.026559356136820925(66 times). However, here in non-rejection emails thank ranks much higher than interests did in the first one thank is in the top 4 of non-rejection emails with a value of 0.006802721088435374(24 times).

The 3rd most frequent word that appears in rejection emails is Software which also makes a lot of sense because most of the jobs he was applying for were software engineering jobs. The frequency value of this was 0.018108651911468814 for rejection (45 times). However, in non-rejection marked emails software ranked 2nd in frequency implying that in non-rejection emails a lot of them were about software and the frequency value was 0.01020408163265306(36 times).

Top 3 results for frequency distribution in non-rejection

In non-rejection emails the top frequency wasn’t actually a word in this case it was punctuation specifically this - The value of this was 0.012188208616780046(43 times). Comparing this to the reject emails this punctuation doesn’t appear at all in rejection emails top 30 list. Potentially I should have removed punctuation as well like I did with stop words.

The 2nd most frequent word that appears in non-rejection emails is Software with a value of 0.01020408163265306(36 times). As mentioned briefly in the rejection email section it was a bit surprising to see software appear in non-rejection emails since these weren’t related to job roles however, he may have subscribed to certain software-based emails affecting this that aren’t necessarily roles he applied for. In rejection email software was the 3rd highest in frequency.

The 3rd most frequent word that appears in non-rejection emails is engineer with a value of 0.007086167800453515(25 times). This for a similar reason to software is a bit surprising because these weren’t necessarily job rejection emails but seem to reference jobs but I may do sentiment analysis to analyse this as it has intrigued me a bit.

Sentiment analysis for reject

Initially I thought sentiment analysis was going to be beneficial and to do this I used the Vader model. However, looking at the results I now understand that it isn’t the most useful thing to analyse here for rejection emails because the language used in the rejection emails try to convey a positive outcome despite you have been rejected which is negative outcome. Which somewhat can confuse the Vader model here I believe that although this data isn’t the most beneficial due to the categorisation of rejection specifically being marked positive it was interesting however, to see that Vader is somewhat tricked by the language used in rejection emails because they all tend to start of saying words like thanks which usually connotes positive feeling. However, the email then uses unfortunately and rejects the application which Vader somewhat doesn’t seem to always consider so some of the results are noted as positive when I believe they probably shouldn’t be in the full context.

When reading the individual emails in non-reject I see that coding interviews are categorised as non-reject emails this could be a large factor in why there is such an overlap in the frequency distribution including words like software, engineer, interest, and others contained in both the top 30 lists due to the fact he included emails related to jobs that aren’t necessarily rejections. This largely answers why the frequency distributions had similar words for reject and non-reject.

Sentiment analysis for non-reject

However, when it comes to non-rejection emails, I feel sentiment analysis performs a lot better here it tends to rank most the emails in correct categories like positive, neutral, negative. I believe this is due to a lot clearer emails being used which doesn’t share the same format as rejection emails which tend to as be mentioned try to spin a rejection into a positive by saying thanks for applying and overall being negative. Whereas the non-rejection emails in this case are very much more concise and clearer in nature which perhaps means Vader performs sentiment analysis far better on these emails as they are less contradictory in their format. It is important to note that using Vader for sentiment analysis means we have a pre trained model that has already been given data to learn from and that is how Vader determines the sentiment of the emails.

Conclusion

Overall, analysing the rejection and the non-rejection emails has been quite interesting. Initially I was planning to only do frequency distribution of words but when I got the results and a lot of the results for rejection and non-rejection were very similar in nature and we can see that in both reject and non-reject certain words appear frequently in both. And due to this fact, I wanted to try and understand what else we could analyse, and I also felt that digging deeper into this would be beneficial for understanding why these emails, despite being categorised as non-reject and reject had overlapping words in the top 30. Also, the sentiment analysis code I did is good because it displays the full email and then whether it was deemed positive or negative or neutral. This was beneficial because in reject emails Vader did somewhat struggle to denote positive, negative, or neutral to these emails I would deem most rejection email negative, but Vader didn’t do this. However, sentiment analysis performed far better when it came to non-rejection emails which I tended to agree more with Vader’s trained model and how it classified the data. The sentiment analysis was useful because it also enables me to read the email then make up my own mind on the sentiment of the specific email in question. I did this for some of the emails when I was reading the sentiment analysis results and making up my own mind also allowed me to question whether I agreed with Vader’s classification of that email.

Appendix

The below code is rejection emails and their frequency distribution I removed stop words because the stop words were making the data considerably less interesting.

from collections import Counter

import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

rejection\_emails = df[df['Status'] == 'reject']

rejection\_tokens = [word for email in rejection\_emails['Email'] for word in email.split() if word.lower() not in stop\_words]

word\_counts = Counter(rejection\_tokens)

total\_words = sum(word\_counts.values())

sorted\_word\_frequencies = sorted(((word, count / total\_words) for word, count in word\_counts.items()), key=lambda x: x[1], reverse=True)

for word, frequency in sorted\_word\_frequencies[:30]:

    raw\_count = word\_counts[word]  # Raw count of the word

    print(f"{word}: {frequency} (Raw count: {raw\_count})")

the below code is non rejection emails and their frequency distribution I removed stop words because the stop words were making the data considerably less interesting.

from collections import Counter

import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

non\_rejection\_emails = df[df['Status'] == 'not\_reject']

non\_rejection\_tokens = [word for email in non\_rejection\_emails['Email'] for word in email.split() if word.lower() not in stop\_words]

counts\_non\_rejection = Counter(non\_rejection\_tokens)

total\_words\_non\_rejection = sum(counts\_non\_rejection.values())

sorted\_word\_frequencies\_non\_rejection = sorted(((word, count / total\_words\_non\_rejection) for word, count in counts\_non\_rejection.items()), key=lambda x: x[1], reverse=True)

for word, frequency in sorted\_word\_frequencies\_non\_rejection[:30]:

    raw\_count = counts\_non\_rejection[word]  # Raw count of the word

    print(f"{word}: {frequency} (Raw count: {raw\_count})")

below code is the code for sentiment analysis of reject emails

import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

nltk.download('vader\_lexicon')

sentimentanalyser = SentimentIntensityAnalyzer()

reject\_emails = df[df['Status'] == 'reject']

for email\_text in reject\_emails['Email']:

    sentiment\_data = sentimentanalyser.polarity\_scores(email\_text)

    print(email\_text)

    print(f"Sentiment scores: {sentiment\_data}")

    # Determine the overall sentiment

    if sentiment\_data['compound'] >= 0.05:

        print("Positive")

    elif sentiment\_data['compound'] <= -0.05:

        print("Negative")

    else:

        print("Neutral")

    print()

below is the sentiment analysis code for non reject emails

import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

nltk.download('vader\_lexicon')

sentimentanalyser = SentimentIntensityAnalyzer()

non\_rejection\_emails = df[df['Status'] == 'not\_reject']

for email\_text in non\_rejection\_emails['Email']:

    sentiment\_data = sentimentanalyser.polarity\_scores(email\_text)

    print(email\_text)

    print(f"Sentiment scores: {sentiment\_data}")

    # Determine the overall sentiment

    if sentiment\_data['compound'] >= 0.05:

        print("Positive")

    elif sentiment\_data['compound'] <= -0.05:

        print("Negative")

    else:

        print("Neutral")

    print()