Sentiment Analysis on tweets about remote work

Problem:
Given the changing landscape of the workplace from office to home, people have different opinions regarding these large scale transitions. Many say this trend would likely continue after the pandemic. However, there will be many challenges that companies will need to overcome in order for remote work to stay. But the question still remains, how do remote workers actually feel ? and what are the challenges that they are facing daily? This insight will help businesses to plan and prioritize their next step in this paradigm shift.

I plan to classify tweets according to their sentiments (positive, negative, or neutral) to answer the first question. I would also like to implement simple topic identification on the negative tweets to get some ideas on the problems people have to face in relation to remote working.

Data:
I will scrape Twitter using twint. There is no limit in the amount of data entries that I can extract. I plan to use 100,000 entries.

Approach:

1. Preprocessing and transformation:
    - Cleaning text by removing stop words and emoji.
2. Text Analysis: Measure length of each text.
3. **Rule based** Sentiment Analysis using **Vader**:
   Vader performs especially well with social media text. Which is exactly what we need for our tweets.
   Rule based approach: The algorithm calculates the sentiment score from a set of manually created rules. For example, you can count the number someone used “great” in their review, and increase the estimated sentiment for each. It sounds very simplistic, but that is basically what’s happening, just on a much larger scale.
4. From the negative assigned tweets, create 3 clusters (positive, negative, neutral) of manually generated words using Gensim ‘most similar’.
5. Custom Sentiment Analysis using **BERT:**
    - Use BERT tokenizer and encoder to embed our target clusters and our tweet text.
    - Use cosine similarity to decide which text goes to which cluster.
6. Use BERT for topic modelling/ keyword extraction by first creating candidate keywords using simple CountVectorizer