Sentiment Classifier Paper

Intro

This project started off super challenging but ended up being something I'm quite proud of. I took a sentiment polarity labeled dataset of amazon food reviews and used them to build a sentiment classifier agent.

Beginning

Compared to previous projects, this dataset felt very flat at first. It only had two columns, the paragraph-long review and the integer indicated polarity.
Despairing, I ran through the steps I knew to do for NLP. It was already cleaned for punctuation/capitalization, so I removed stopwords and stemmed it. From there, I built vectors for tokens, bi-grams, and tri-grams. I naively thought I could dump all this data into a model and get a decent result. I struggled to come up with features to build or steps to take.
Though, when I realized that the combined vector of all of my n-grams was nearly 19 million columns, my conception of what I was doing changed dramatically. I wasn't taking a given dataset and building features on top of it to increase the amount of information present; rather, I was slicing up the immense wealth of information provided in the paragraph-long reviews and trying to extract the smallest number of dimensions that would allow a model to differentiate polarity.

Middle

I realized that there were essentially two challenges ahead of me:

- What level of dimensionality did I actually have to work with?
- What minuscule slice of the data did I want to fill that to be determined dimensional space with?

The first problem was simply empirical. Given my hardware limitations and the project scope, I decided that I would hunt for a 10-minute training time. Given that, I set up a function to repeatedly model the dataset, increasing the number of features it was using each time. I didn't track anything but execution time here. After quite a bit of testing, it came out to be about 600 features for my 650k row dataset. Almost square!
The second was a little bit more challenging. I am a fluid English speaker, often told I have a silver tongue, but my formal understanding of the language is minuscule. So again, this became an empirical exercise rather than one of theory. I knew I was looking to differentiate between my labels, so I decided to do some basic statistics to pave the way. I developed three different lexicons to slice and feed as a vocabulary to my tokenizer; words that only showed up on the positive side, words that only showed up on the negative side, then a list of words that appear

on both ordered by their ratio. I favored words with a high ratio, indicating a strong skew and, thus, hopefully, predictive power.
After some discussion with my mentor, I threw out my bi-gram and tri-grams and instead went with a topic modeling algorithm known as Latent Dirichlet Allocation (LDA). The premise is that it would capture more information than the bi or tri-grams, occupying far less of my rarified dimensions.
Now, I had the dimensional scope constraint established and the repository of dimensions I would draw from to fill out that scope. Again this was a space in which I possessed little to no domain knowledge, so I opted to go for an empirical approach and explore different combinations of the above vectors. Due to time constraints, I didn't do an exhaustive or even bayesian search; I just intuitively chose some values that seemed to map sectors of the possibility space.

End

Results:
I was surprised to find that single polarity only dimensions(words) seem to have very little impact on the model. My guess is that this is because if they're only showing up in one polarity, they're not common enough to be indicators. They'll be sure-fire indicators that a few documents are a certain polarity, but that's it. Generalizability is poor.
I was unsurprised but pleased to find that highly skewed dimensions(words) seem to significantly impact the model. This makes sense to me; by showing up in both polarities, there's a degree of prevalence being enforced, and the high skew ensures they'll have *some* predictive power.
I was surprised and disappointed by topic dimensions being a bit muddled in their impact but seeming to be less effective. I suspect that this will become more useful as the number of documents goes up and the number of topics as well. I would love to explore that with better hardware.
Overall the impact of varying input parameters only caused a .02 difference in recall and a 0.17 difference in precision. So it seems we only have minimal effect on the model results as we shift the inputs. However, my suspicion is that as we scale up the model, that difference will become increasingly important.
Because this model is so dependent upon high-skew words, I doubt it'll be particularly generalizable. It falls apart as soon as we move to a slightly different domain or get documents from people with even a mildly different lexical culture. I think this could be mollified somewhat with the topic modeling, but there's more to see.

Outro
This process was a learning experience at every level. In the end, I have a functional model, but I have so many more things I would like to do to improve it. What follows is an incomplete list of such things:

1. A more exhaustive search of the input parameter space described by our dimensionality constraint

2. Instantiate a local spark network so I can scale up a bit more/deal with the fragility of the jupyter notebook (I crashed it so, so many times)
3. Apply a neural net
    a. Their epoch system would probably allow me to hit larger scales as well.
    b. I suspect I could get more accuracy with one of these as there's a lot of implicit feature generation for it to do.
4. I had three more labeled datasets, and I would have loved to include those, but I was already maxing out on dimensionality with my single dataset.
5. I really would love to have explored more n-grams. Now that I have established my lexical slicing pattern, some exploration could be done there.
    a. I also suspect the slicing pattern could be productive if mixed with the n-gram association vectors. Lots to explore...
6. Better/more hyper parameter testing, cross-validation, and a bunch of other little improvements
    a. I had many of these in place, but near the end, I was madly stripping away complexity to get things to run on my local machine.
7. I would love to test my model on other labeled datasets. I suspect it has poor generalizability, but I don't know yet.