Collin Daley

Thomas Duncan

Zach Ellis

Brad Jensen

Kyle Miller

NLP Analysis of Amazon Product Reviews

An important practice for anyone trying to create a product is understanding how similar products do in the same market. For this purpose, we have decided to use Natural Language Processing on reviews of 60 different sellers, selling a version of the same product. This product was Phosphatidylserine, a supplement for brain functionality and sleep. We found that the positive sentiment towards the product was generally about the base product, the supplement doing well for the body. The negative sentiments were generally about aftertaste and delivery. There are business opportunities for someone trying to sell this product as well. We will describe our process and results below.

The first step in the project was gathering the data related to our keyword from Amazon’s online storefront and formatting it all to be saved into a pandas dataframe. This was an important first step for the project, since it would be very difficult to design and test the NLP aspects of the project without a sizable dataset to draw from. It was unfortunate, then, that this step had numerous complications that made its completion difficult. The major complication being that Amazon dislikes automated web programs that gather data from their storefront, and so blocks them whenever they are detected. This meant that basic web-scraping designs would not work for our needs, and a more sophisticated scraping technique had to be developed to prevent our script from being blocked.

The main idea behind web-scraping Amazon is making the script appear as human-like as possible. Basic web-scrapers skip over many inefficient parts of web-browsing as a way to improve runtime and reduce the complexity of developing them, but these shortcuts are dead giveaways to any anti-scraping defense system. And so, although it massively increases the runtime of the program, the most significant change that can be made to appear more human is simply slowing the script down. Random time delays were added between all requests to Amazon’s servers, which effectively prevented the first layer of alarms from being raised.

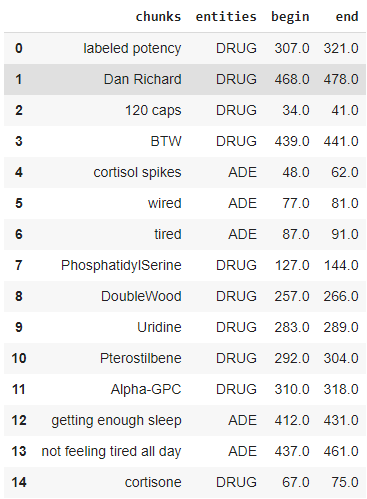
However, time delays are not enough. Realistic and infrequently changing network headers are required as well. These headers provide Amazon with a slurry of information about the web browser making the request to their webpage, and by injecting real headers taken from our own machines into the requests made to Amazon, we can create the illusion that our requests coming from a human-used web-browser, rather than an automated, browser less script.

The final step in the process is the most complicated, yet most effective. Due to the aforementioned shortcuts taken by most scrapers for the sake of efficiency, they have a very inhuman behavior that’s easy to detect even with time delays and realistic headers; that being, scrapers make a request to a specific URL with no prior activity multiple times in a row. A human navigating a webpage will have numerous indications that they came from previous pages, such as cookies, before making a new request, and these indicators can be used to verify human activity. So, to appear human to Amazon’s web servers, the scraper must not treat each request made as a brand-new request made with no prior activity. It must instead remember cookies from previous pages it visited and present them upon making a new request, as well as only making requests to URLs that can be reached via a link on the previously requested page.

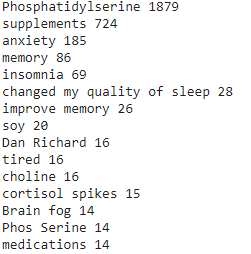
A combination of these three steps, however, is enough to appear human to Amazon’s anti-scraping measures. Due to the inefficiency of these added steps, the script took multiple days to run to completion, but it did run without much further complication. And with the data gathered, it was formatted into a Pandas dataframe, ready to be used for the next steps of the project.

After we finish web scraping, we need to preprocess the data to remove any unwanted data from our reviews. The types of things that we are removing in this step include URLs, hashtags, emojis, punctuation, symbols, and digits. This step is easy to implement but hard to know exactly what to remove from the dataset, as there are not any standardized things to look for and remove. Once we have removed everything that we do not want in the dataset, we tokenize the reviews to split the reviews up into a list of words. One this step is done; we can remove any stop words from our data. We remove these words because they do not add much meaning to an overall sentence. Because of this, we can safely remove and ignore these words without removing the meaning of the sentence. The final step that we take during the preprocessing step in the process is lemmatizing each remaining word in the review. Lemmatizing a word is essentially the process of removing prefixes and suffixes from words so that the word is reduced to its simplest form, known as its stem. This shortens our vocabulary and dictionary needed to process the data, improving speed and efficiency of your program. Finally, when we finished preprocessing the reviews, we saved each set of reviews in a new row of our dataframe as a list of lists. Before the preprocessing step, we had 602 thousand words and 2.9 million characters. After we finished this step, we were down to 265 thousand words and 1.5 million characters. We removed nearly half the data that had been originally scraped from Amazon during the last step.

With all the reviews gathered and put into a dataframe, we can perform some named entity recognition on them. Getting the top or most common entities can give the most important ideas or points that all the reviews say. With these key entities we can somewhat reverse engineer what these entities refer to or what it does. The first step is to find a pretrained model to do the NER. After some trial and error, we ended up choosing Spark’s Adverse Drug Event NER. This model seemed to get a good range of entities both clinical and other. Some filtering of the entities was needed as the recognizer returned some types and groups of entities that didn’t provide any useful information.



After running the reviews through the NER model we got a big dataframe with all the entities found. There are thousands and thousands of entities in the dataframe so we needed to process them to find the most important or most common. To do this we used Spacy’s similarity feature to compare chunks. If two chunks were similar enough, they were merged into one chunk that was mentioned twice. Using this we combine all the thousands of entities to a list of most to least common chunks. While Spacy does a good job in finding and combining these chunks, it isn’t as comprehensive as Spark and doesn’t recognize some of the chunks. There is room for improvement here along with some more filtering for our specific case. With the list of most common chunks, we simply take the top 15 to analyze.



The final step was to perform Topic Modeling on the data. Topic Modeling is a type of statistical modeling for discovering the abstract “topics” that occur in a collection of documents. The goal was to take the reviews and use Latent Dirichlet Allocation and Term Frequency - Inverse Document Frequency for the topic modeling.

We will explain the process shortly and concisely. First we received the preprocessed data in the form of a pickle file. We then transferred that to a Pandas Dataframe. From there the next step is to use Gensim to create a dictionary of the words used in the reviews.

We then filter out our extremes, this step is necessary for saturation purposes. Next, using Gensim’s doc2bow function, we create a bag of words (BoW) of the words used in the reviews. This also records how many times the word is repeated in the reviews. We then use Gensim to create a TF-IDF and LDA model on the Bag of Words for the reviews.

Finally, we overlay the TF-IDF on the LDA bag of words model. This creates the model necessary for Topic Modeling.

[part 2]

With all the basic components of the project completed, we were able to draw some conclusions about the keyword we were given. The topic modeling process provided us with a series of topics that surround the general keyword when searched on Amazon’s storefront, and from this we could identify the key features of the product. We concluded that phosphatidylserine is the primary chemical used in a type of supplement pill intended to aid general brain activity. The pill is derived from fish oil, and the fishy taste and smell of the pills is a common complaint amongst reviewers. Positive reviews of the pill report improvements in a wide variety of mental faculties. Reviewers report that phosphatidylserine pills improve their focus, aid in restful sleep, increase their memory recall, and a general increase in mental clarity throughout the day.

The results for the Topic Modeling confirm the results of the sentiment analysis. As you can see below, in the list of word clouds of Topic 0 - 9, there were common topic threads that share ground with the sentiment analysis. The most important thing to take from the Topic Modeling is that the product has positive effects on the brain and sleep and has poor aftertaste.

In conclusion the features that matter to customers and buyers of Phosphatidylserine are the effects it has on their bodies, taste, and delivery. People disliked the aftertaste and some delivery issues. There are business opportunities if one would like to pursue it. If one could make the same product with a good or even neutral aftertaste it could perform well. This was a powerful display of how Natural Language Processing can help a business in many ways.