USING NLP TO PRESENT A DATA

Let's look at an example or customer reviews collected for a product. In this example, we will see top reasons for positive reviews and negative reasons.

What is Natural Language Processing (NLP)?

Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data. Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural language generation. The development of NLP applications is challenging because computers traditionally require humans to "speak" to them in a programming language that is precise, unambiguous and highly structured, or through a limited number of clearly enunciated voice commands. Human speech, however, is not always precise -- it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.

This topic is worth a complete book on itself. In this section, we will see an example how we can use the NLP concept to analyze a text data (Customer Reviews). We will analyze the review_comment data in the order_reviews.csv and we will pick top 3 reasons why customers like or dislike a product. We will learn how to process raw text step by step.

Code:

Step 1: Prepare the data for the analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

######### Step 1: Data Preprocessing
#Read the csv files - orders
order_df =
pd.read_csv('https://raw.githubusercontent.com/swapnilsaurav/OnlineRetail/master/orders.csv')
#Display all the column names
print(list(order_df.columns))

# Convert columns to datetime
order_df['order_purchase_timestamp'] =
pd.to_datetime(order_df['order_purchase_timestamp'])
order_df['order_delivered_customer_date'] =
pd.to_datetime(order_df['order_delivered_customer_date'])

#Read the csv files order_reviews
order_rev_df =
pd.read_csv('https://raw.githubusercontent.com/swapnilsaurav/OnlineRetail/master/order_reviews.csv')
#Display all the column names
```

```
print(list(order_rev_df.columns))

# Convert columns to datetime
order_rev_df['review_creation_date'] =
pd.to_datetime(order_rev_df['review_creation_date'])
order_rev_df['review_answer_timestamp'] =
pd.to_datetime(order_rev_df['review_answer_timestamp'])

#Merge Orders and Reviews
reviews = pd.merge(order_df, order_rev_df, on='order_id',how='left')
wehavecount = reviews['order_id'].count()

# Remove unused columns
to_drop = [
    'review_id',
    'order_id',
    'customer_id',
    'review_comment_title',
    'order_approved_at',
    'order_delivered_carrier_date',
    'order_estimated_delivery_date'
]
reviews.drop(columns=to_drop, inplace=True)
```

Output of Step 1:

```
['order_id', 'customer_id', 'order_status', 'order_purchase_timestamp', 'order_approved_at',
 'order_delivered_carrier_date', 'order_delivered_customer_date', 'order_estimated_delivery_date']
['review_id', 'order_id', 'review_score', 'review_comment_title', 'review_comment_message',
 'review_creation_date', 'review_answer_timestamp']
```

Step 2: Plot graphs to analyze the data

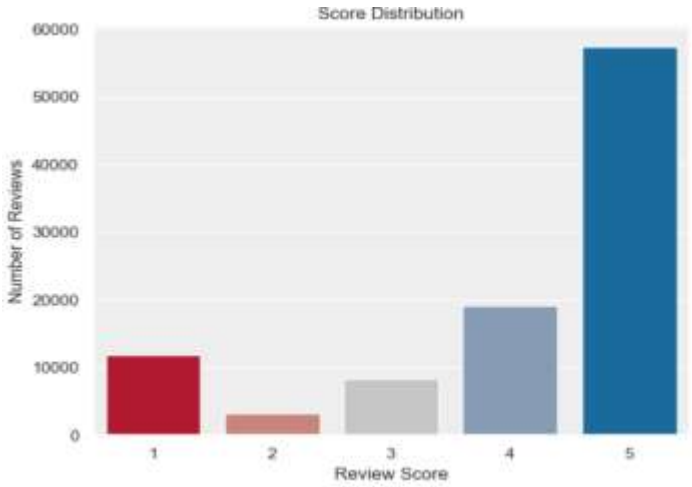
```
######### Step 2: Plots to understand the dataset
from datetime import datetime
sns.set()
# 5Star: BLUE to 1 Star RED
COLOR_5S = '#0571b0'
COLOR_1S = '#ca0020'
REVIEWS_PALETTE = sns.color_palette((COLOR_1S, '#d57b6f',
'#c6c6c6', '#7f9abc', COLOR_5S))
# White background
sns.set_style('darkgrid', {'axes.facecolor': '#eeeeee'})
# Default figure size
resize_plot = lambda: plt.gcf().set_size_inches(12, 5)

p_5s = len(reviews[reviews['review_score'] == 5]) * 100 /
len(reviews)
p_1s = len(reviews[reviews['review_score'] == 1]) * 100 /
len(reviews)
```

```
first_dt = reviews['review_creation_date'].min()
last_dt = reviews['review_creation_date'].max()
avg_s = reviews['review_score'].mean()
print(len(reviews), 'reviews')
print('First:', first_dt)
print('Last:', last_dt)
print(f'5Star: {p_5s:.1f}%')
print(f'1Star: {p_1s:.1f}%')
print(f'Average: {avg_s:.1f}')

# Score Distribution as Categorical Bar Graphs
sns.catplot(
    x='review_score',
    kind='count',
    data=reviews,
    palette=REVIEWS_PALETTE
).set(
    xlabel='Review Score',
    ylabel='Number of Reviews',
);
plt.title('Score Distribution')
plt.show()

#Review Created Date Compared to Purchase Date
reviews['review_creation_delay'] =
(reviews['review_creation_date'] -
reviews['order_purchase_timestamp']).dt.days
sns.scatterplot(
    x='order_purchase_timestamp',
    y='review_creation_delay',
    hue='review_score',
    palette=REVIEWS_PALETTE,
    data=reviews
).set(
    xlabel='Purchase Date',
    ylabel='Review Creation Delay (days)',
    xlim=(datetime(2016, 8, 1), datetime(2018, 12, 31))
);
resize_plot()
plt.title('Review Created Date Compared to Purchase Date')
plt.show()

#Reviews by month using the order_purchase_timestamp column and
plot a timeseries. Consider reviews created after purchase date
# Review group by Month
reviews['year_month'] =
reviews['order_purchase_timestamp'].dt.to_period('M')
reviews_timeseries = reviews[reviews['review_creation_delay'] >
0].groupby('year_month')['review_score'].agg(
    ['count', 'mean'])
```

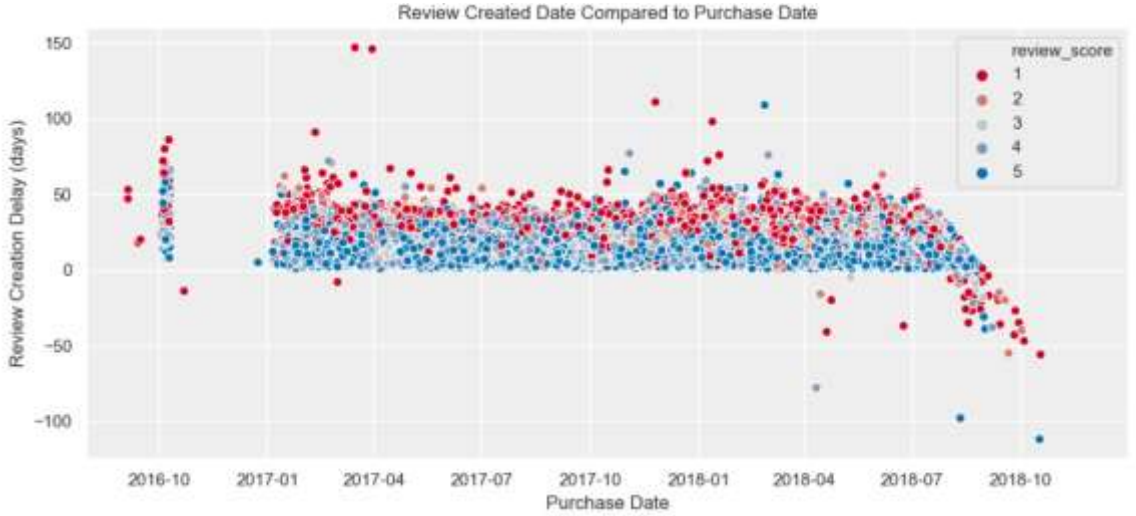
```
ax = sns.lineplot(
    x=reviews_timeseries.index.to_timestamp(),
    y='count',
    data=reviews_timeseries,
    color='#984ea3',
    label='count'
)
ax.set(xlabel='Purchase Month', ylabel='Number of Reviews')
sns.lineplot(
    x=reviews_timeseries.index.to_timestamp(),
    y='mean',
    data=reviews_timeseries,
    ax=ax.twinx(),
    color='#ff7f00',
    label='mean'
).set(ylabel='Average Review Score');
resize_plot()
plt.title("Review group by Month")
plt.show()

#Exploring Review Comments
reviews['review_length'] =
reviews['review_comment_message'].str.len()
reviews[['review_score', 'review_length',
'review_comment_message']].head()

#Size of the Comments
g = sns.FacetGrid(data=reviews, col='review_score',
hue='review_score', palette=REVIEWS_PALETTE)
g.map(plt.hist, 'review_length', bins=40)
g.set_xlabels('Comment Length')
g.set_ylabels('Number of Reviews')
plt.gcf().set_size_inches(12, 5)
plt.title("Size of the Comments")
plt.show()

#Review Size and the Rating
ax = sns.catplot(
    x='order_status',
    kind='count',
    hue='review_score',
    data=reviews[reviews['order_status'] != 'delivered'],
    palette=REVIEWS_PALETTE
).set(xlabel='Order Status', ylabel='Number of Reviews');
plt.title("Order Status and Customer Rating")
plt.show()
resize_plot()
```

Output of Step 2:

```
100000 reviews
First: 2016-10-02 00:00:00
Last: 2018-08-31 00:00:00
5Star: 57.4%
1Star: 11.9%
Average: 4.1
```

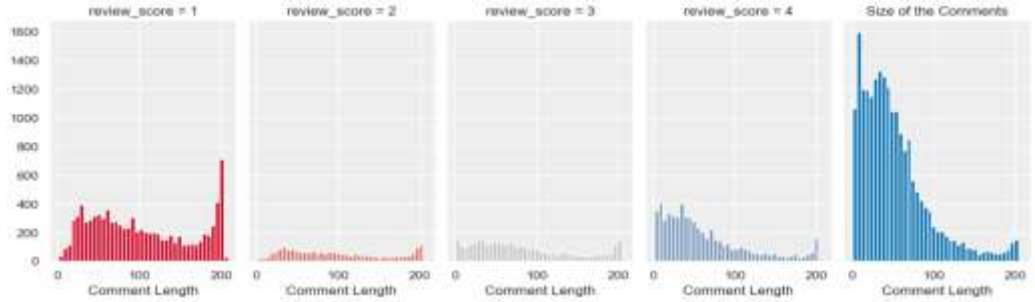


Maximum reviews are 5 Star. It is interesting to observe that there's more 1 star reviews than 2/3 stars reviews.

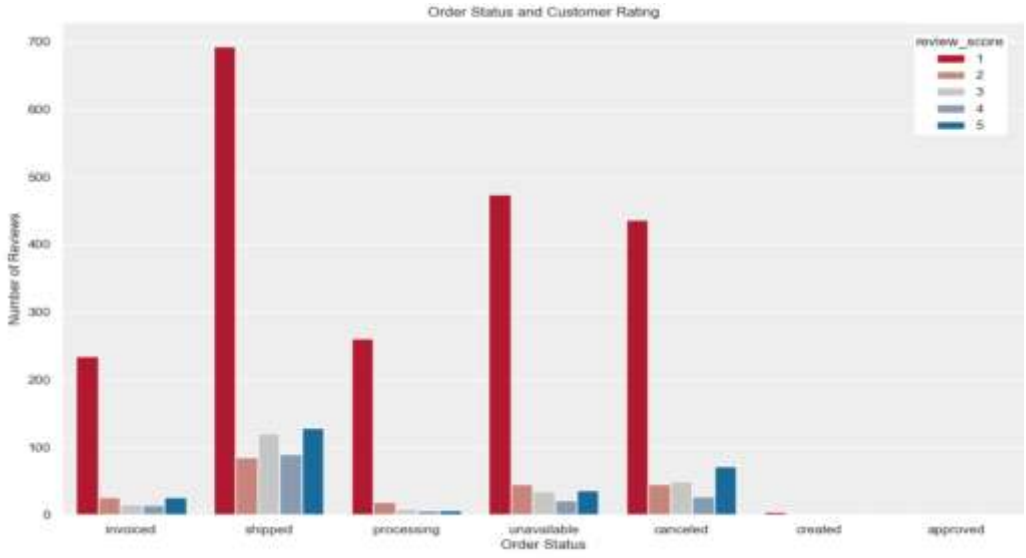


The graph shows the spread of various reviews given from date of purchase. There are few reviews (mostly in October of 2018) shows reviews were given before purchase date. This could be because of error in the data.

Here we group reviews by month using the order_purchase_timestamp column and plot a timeseries. We will only consider reviews created after the purchase date here. There are 2 lines – Mean line talks about the average reviews at the given point in time. Overall score was close to 5 in December of 2016 but it fell to below 4 in early 2018 but since then it has improved to go over 4. The other line shows the total count of the reviews. We see that there was a big jump in the number of reviews given during November and December of 2017.



Customers tend to write lengthier reviews when they are not satisfied.



If we plot the review score distribution of orders that do not have a 'delivered' status, we can see that most of them have a 1 star rating.

Step 3: Perform NLP Analysis

Following steps have been followed here:

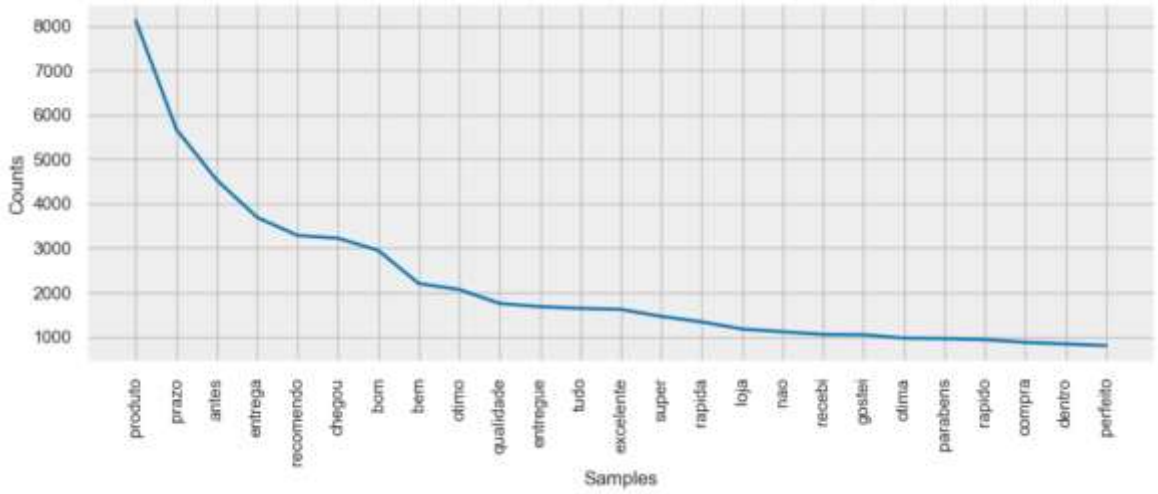
- Convert text to lowercase
- Compatibility decomposition (decomposes ã into a~)
- Encode to ascii ignoring errors (removes accents), reencoding again to utf8
- Tokenization, to break a sentence into words
- Removal of stop words and non-alpha strings (special characters and numbers). A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We will remove words from our analysis as they dont give vital information.
- Generally next step we perform would have been Lemmatization (transform into base or dictionary form of a word). Lemmatization is not available for Portuguese words with the NLTK package so we will ignore that in this case
- N-grams creation (group lemmas next to each other, by comment)
- Grouping n-grams of all comments together. An N-gram means a sequence of N words.

```
import unicodedata
import nltk
###### Step 3: Perform Following functions are required to run NLP
#3.1 Remove accept / local dialect
def remove_accents(text):
    return unicodedata.normalize('NFKD', text).encode('ascii',
errors='ignore').decode('utf-8')

#3.2 Remove stop words in Portuguese
STOP_WORDS = set(remove_accents(w) for w in
nltk.corpus.stopwords.words('portuguese'))
STOP_WORDS.remove('nao')  # This word is key to understand delivery
problems later

#3.3 Tokenize the comment - break a sentence into words
def comments_to_words(comment):
    lowered = comment.lower()
    normalized = remove_accents(lowered)
    tokens = nltk.tokenize.word_tokenize(normalized)
    words = tuple(t for t in tokens if t not in STOP_WORDS and
t.isalpha())
    return words

#3.4 Break the words into unigrams, bigrams and trigrams
def words_to_ngrams(words):
    unigrams, bigrams, trigrams = [], [], []
    for comment_words in words:
        unigrams.extend(comment_words)
        bigrams.extend(' '.join(bigram) for bigram in
nltk.bigrams(comment_words))
        trigrams.extend(' '.join(trigram) for trigram in
nltk.trigrams(comment_words))
```

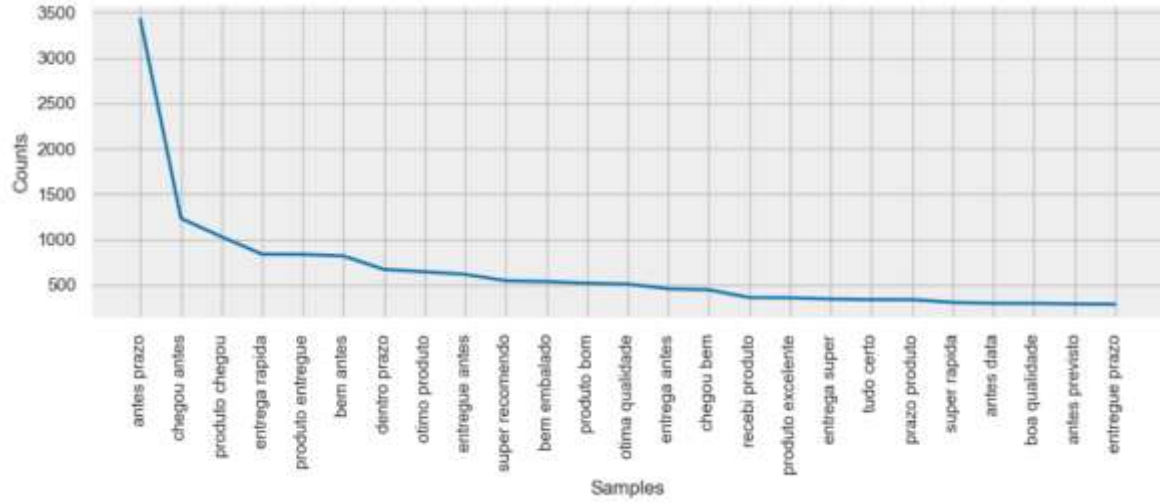
```
    return unigrams, bigrams, trigrams

def plot_freq(tokens, color):
    resize_plot = lambda: plt.gcf().set_size_inches(12, 5)
    resize_plot()
    nltk.FreqDist(tokens).plot(25, cumulative=False, color=color)

#Now go ahead with analysis
sns.set()
# 5Star: BLUE to 1 Star RED
COLOR_5S = '#0571b0'
COLOR_1S = '#ca0020'
REVIEWS_PALETTE = sns.color_palette((COLOR_1S, '#d57b6f', '#c6c6c6',
'#7f9abc', COLOR_5S))
# White background
sns.set_style('darkgrid', {'axes.facecolor': '#eeeeee'})
# Default figure size
resize_plot = lambda: plt.gcf().set_size_inches(12, 5)

commented_reviews =
reviews[reviews['review_comment_message'].notnull()].copy()
commented_reviews['review_comment_words'] =
commented_reviews['review_comment_message'].apply(comments_to_words)

reviews_5s = commented_reviews[commented_reviews['review_score'] ==
5]
reviews_1s = commented_reviews[commented_reviews['review_score'] ==
1]

unigrams_5s, bigrams_5s, trigrams_5s =
words_to_ngrams(reviews_5s['review_comment_words'])
unigrams_1s, bigrams_1s, trigrams_1s =
words_to_ngrams(reviews_1s['review_comment_words'])


#Now we will perform NLP analysis to understand it better
#Step 1: frequency distributions for 5 star n-grams
plot_freq(unigrams_5s, COLOR_5S)
plot_freq(bigrams_5s, COLOR_5S)
plot_freq(trigrams_5s, COLOR_5S)

#Step 2: Frequency distributions for 1 star n-grams
plot_freq(unigrams_1s, COLOR_1S)
plot_freq(bigrams_1s, COLOR_1S)
plot_freq(trigrams_1s, COLOR_1S)
```

Output of Step 3:

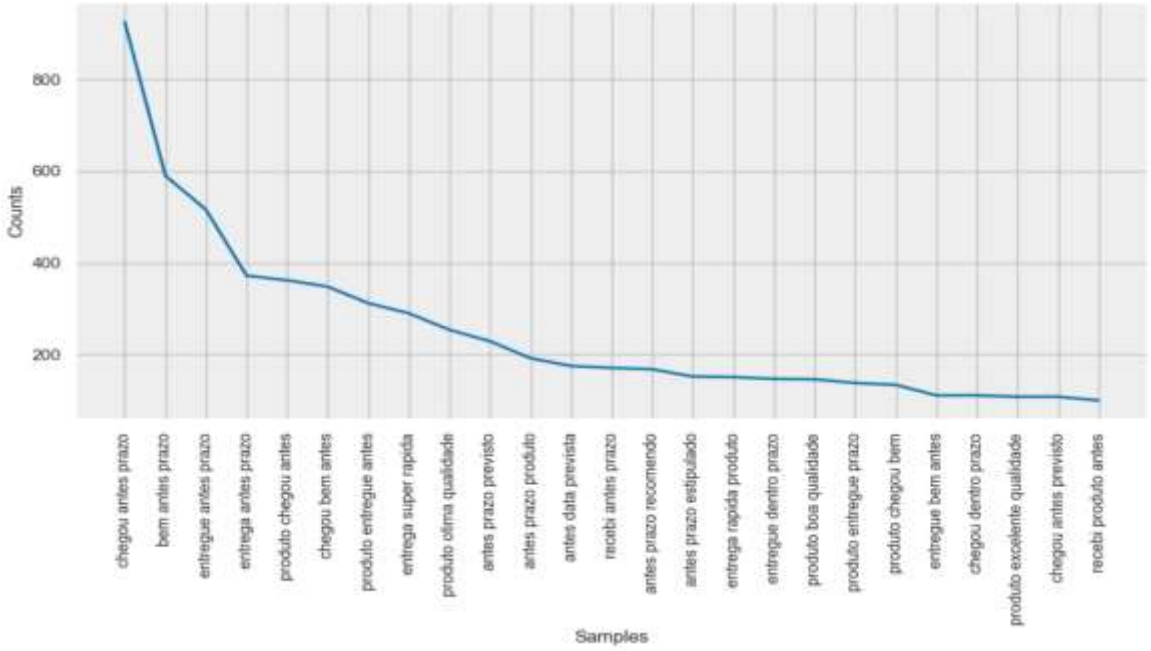
Review 5 Unigram:



Review 5 Bigram:



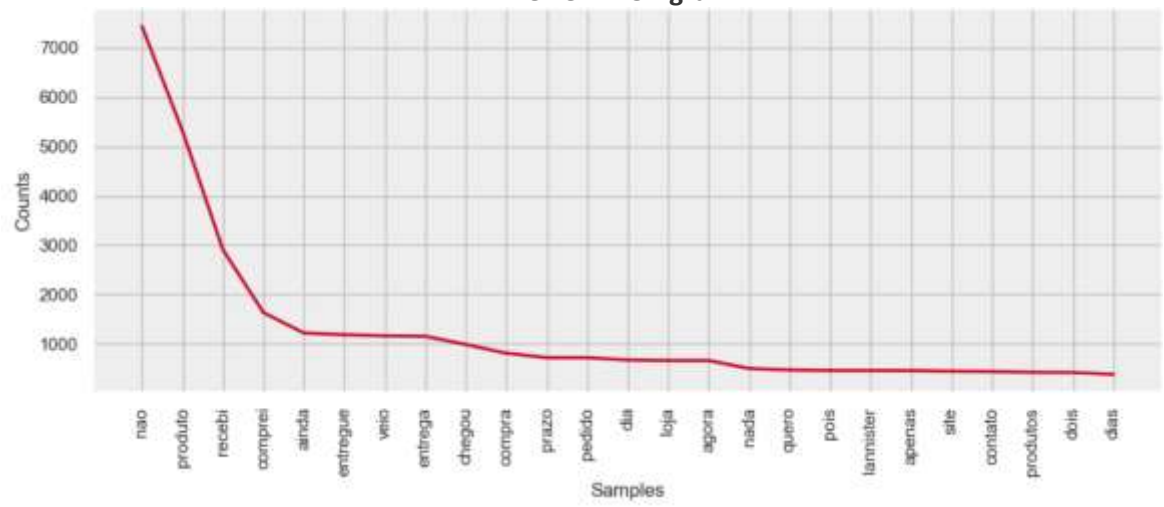
Review 5 Trigram:



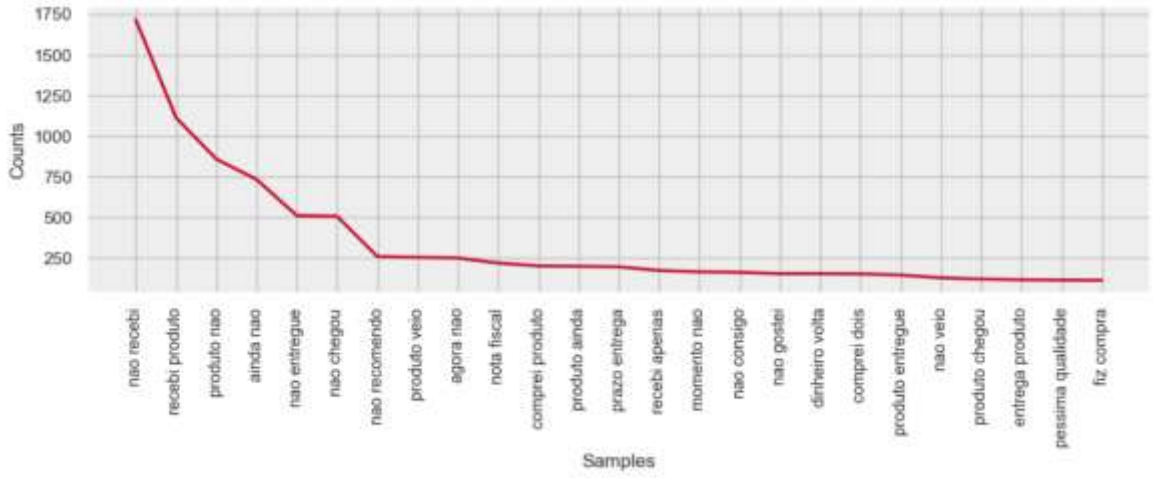
Below are the frequency distributions for 5 star n-grams. We can identify some key topics customers enjoy about their experience:

- Fast delivery ('chegou antes prazo', 'entrega rapida', 'entregue antes prazo', 'super rapida')
- High quality goods ('produto otima qualidade', 'otimo produto', 'produto excelente', 'produto boa qualidade')
- Good packaging ('bem embalado', 'produto chegou bem')

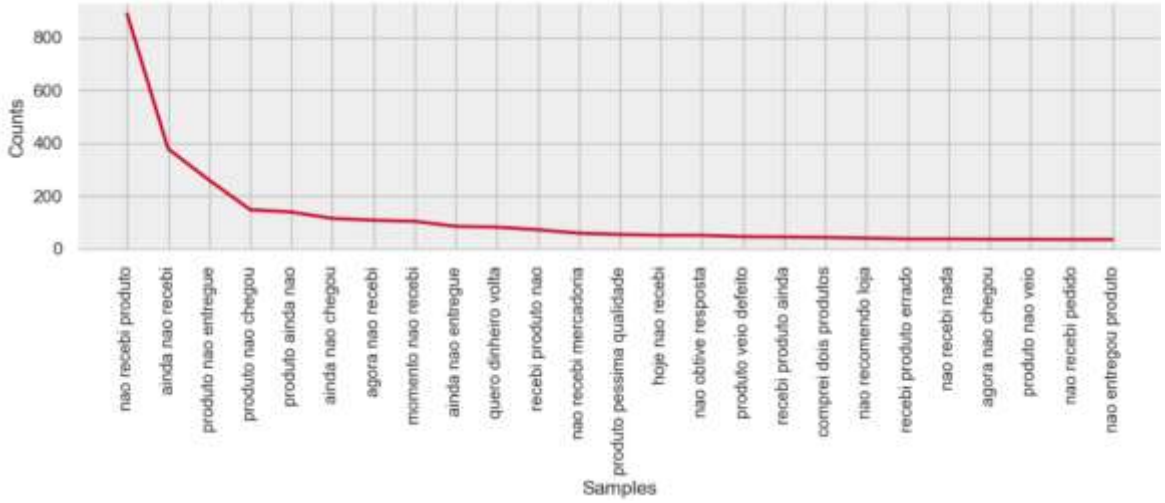
Review 1 Unigram



Review 1 Bigram:



Review 1 Trigram:



Below are the frequency distributions for 1 star n-grams. We can identify some key topics customers dislike about their experience:

- They didn't receive their goods yet ('recebi produto', 'ainda nao recebi', 'produto nao entregue', 'produto nao chegou', 'nao recebi mercadoria')
- They want refund ('quero dinheiro volta')
- Bad quality goods ('produto pessima qualidade', 'produto veio defeito')
- They had some problem when purchasing 2 products ('comprei dois produtos')