UNIVERSITÀ POLITECNICA DELLE MARCHE
FACULTY OF ENGINEERING
Department of Information Engineering
Master in Computer and Automation Engineering.

DATA SCIENCE COURSE EXAMINATION PROJECT

Combine sentiment analysis and topic modeling to cluster Amazon customer reviews

Authors

Valerio Morelli
Federica Paganica
Federico Staffolani
Enrico Maria Sardellini

ACADEMIC YEAR 2024-2025

Contents

I The BERTopic framework **1**

1 The topic modeling task **2**
1.1 Project Purpose . . . 2
1.2 What is Topic Modeling? . . . 2
1.2.1 The state of the art in Topic Modeling . . . 2
1.2.2 The importance of contextual semantic . . . 3
1.3 The pipeline of the BERTopic framework . . . 3
1.3.1 The embedding phase . . . 3
1.3.2 The dimensionality reduction phase . . . 4
1.3.3 The clustering phase . . . 4
1.3.4 The topic extraction phase . . . 5
1.3.5 The post-processing phase . . . 6

II Analysis implementation **7**

2 Scraping and visualizing reviews **8**
2.1 Scraping Amazon reviews . . . 8
2.1.1 Getting the URL . . . 9
2.1.2 Extract the text of the reviews . . . 9
2.2 Visualizing the ratings trend over time . . . 10
2.3 Plotting Word Clouds . . . 10
2.4 Distribution of the lengths of the reviews . . . 11

3 Topic Modeling with BERTopic **13**
3.1 Data pre-processing . . . 13
3.1.1 Long 5-star reviews are sometimes paid . . . 13
3.1.2 Some reviews are very long . . . 13
3.1.3 Some of the reviews are outliers . . . 13
3.2 Post-processing operations . . . 14
3.2.1 Merging similar topics . . . 14
3.2.2 Naming the topics . . . 14

3.3 Understanding the results 14
3.3.1 Calculating the topic distribution 14
3.3.2 Visualizing the topics 18
3.3.3 Why is this useful? 19

4 Sentiment analysis with TextBlob **21**
4.1 TextBlob for sentiment analysis 21
4.2 Calculation of the topic sentiments 21
4.3 Visualizing the sentiments 21

Bibliography **23**

List of Figures

1.1 The pipeline of the BERTopic framework chosen for this project. (Image by the authors) . . . 4
1.2 *HDBSCAN*, unlike *DBSCAN*, is able to detect clusters with different densities. On the left, *HDBSCAN* correctly finds 3 clusters and a set of outliers. (Image by the authors) . . . 5
1.3 The three components of the *KeyBERT* model. (Image by the authors) . . . 6

2.1 The product whose reviews we are going to use for analysis. (Image from Amazon.com) . . . 8
2.2 The history of the HP printer shows a rather stable price over the last two years. (Image by the authors) . . . 9
2.3 The number of monthly reviews. . . . 10
2.4 The distribution of stars in the reviews on a quarterly basis. The red rectangle in the figure shows the increasing polarization of customer opinions in recent years. . . . 11
2.5 The word cloud of the 5 stars reviews. . . . 11
2.6 The word cloud of the 1 stars reviews. . . . 12
2.7 Looking at the distribution of review length, some reviews are very short and some are very long compared to the rest. . . . 12

3.1 The 40 extracted, post-processed topics. ➪ **The interactive version can be accessed here.** . . . 15
3.2 The map of the 40 topics on a reduced 2-dimensional embedding space. The size of the circles is proportional to the frequency on that topic. ➪ **The interactive version can be accessed here.** . . . 16
3.3 The similarity matrix of the topics shows how much the topics are related — how close they are in the reduced embedding space. ➪ **The interactive version can be accessed here.** . . . 17
3.4 The most likely topics of Jerry's comment. ➪ **The interactive version can be accessed here.** . . . 18
3.5 The approximated distribution over the Jerry's sentence. ➪ **The interactive version can be accessed here.** . . . 18

3.6 The map of all the 2.821 extracted from the 500 reviews. Each island contains customers' opinions about a particular aspect of the printer. ➪ **The interactive version can be accessed here.** 19
3.7 Three samples extracted from different topic in the map. ➪ **The interactive version can be accessed here.** 20

4.1 The topics are plotted in a polarity x subjectivity matrix. The topics farthest to the left are the most negative, while the topics farthest to the right have more positive than negative reviews. 22

Part I

The BERTopic framework

CHAPTER 1

The topic modeling task

This chapter introduces the project's goals and explores the definition of topic modeling. Section 1.1 outlines the project's purpose, while Section 1.2 delves into the meaning of topic modeling. Section 1.3 briefly explains the pipeline of the BERTopic framework.

1.1 Project Purpose

This project combine *sentiment analysis* and *topic modeling* to discover which aspects of a product are driving the overall customers' perception. Starting from a famous Amazon product — one that has a lot of reviews spanning across several years — we want to identify "hot topics" in customer reviews to understand product weaknesses and strengths.

Furthermore, after identifying the topics, in order to enrich the final overview of the product, we will classify them according to their *positivity* and *subjectivity*.

1.2 What is Topic Modeling?

Topic Modeling, is a type of statistical modeling that uses *unsupervised* Machine Learning to identify clusters or groups of similar words within a body of text. This text mining method uses semantic structures in text to understand unstructured data without predefined tags or training data [Sheridan, 2024].

1.2.1 The state of the art in Topic Modeling

The classical algorithm to solve the *Topic Modeling* task is LDA. LDA uses the *Bag of Words* embedding to count the frequency of words in the documents. In its base version, **LDA is not contextual aware**.

In 2016, Christopher Moody, in its paper *Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec* [Moody, 2016], devised `lda2vec` , a tool that extends standard LDA with *Word2Vec* embedding **to capture static semantics**.

In 2020, Dimo Angelov published the paper *Top2Vec: Distributed Representations of Topics* [Angelov, 2020]. The paper introduces a new method that changes the prospective of the LDA probabilistic algorithm. The *Top2Vec* model basically performs clustering downstream of the common document and word semantic embedding. In its original form, however, it did not take advantage of modern transformers and was thus limited to static semantics.

In 2022, Maarten Grootendorst authored the paper *BERTopic: Neural topic modeling with a class-based TF-IDF procedure* [Grootendorst, 2022], releasing `BERTopic` . The idea behind BERTopic is to use modern BERT-like transformers instead of the less recent *Word2Vec*, *Doc2Vec* and *Bag of Words* embeddings.

1.2.2 The importance of contextual semantic

BERT transformers excel in capturing **contextual semantic**. This means that the same word, in two different contexts, with two different meanings — e.g. the "bank" that holds your money and the "bank" of a river — **is assigned to two different embedding vectors**. In the language of transformers, this is called the *attention mechanism*.

More specifically, Sentence Transformers, also known as *SBERT*, are what are used here. SBERT models extend the basic BERT models with the ability to embed entire documents along with their individual words.

1.3 The pipeline of the BERTopic framework

BERTopic defines a precise workflow, divided into 4 main phases. The `BERTopic` class adopts the *CBDP* philosophy: in each phase we can use one between different components. In this project, we used the pipeline shown in figure 1.1.

1.3.1 The embedding phase

The documents in the corpus — in our case, the 500 reviews — are encoded into low-dimensional dense vectors using a transformer. Any model from the *SBERT* library can be used to do so. The *SBERT* documentation contains a table that shows the different available models, ranked by their performance and size.

One thing we should be aware of is the *Max Sequence Length* model attribute. If we have documents that are too long, the content will be truncated and potentially useful information will be lost. **So we can either split the documents into smaller documents**[1] — that's what we did here — **or choose a larger model**.

Another aspect to take into account is the language. If our corpus contains a lot a significant amount of words or even entire documents in a non-english language, we should use a multi-lingual model, like `paraphrase-multilingual-mpnet-base-v2` . However, this is not the case in this project because we scraped the reviews from Amazon US.

We chose `all-mpnet-base-v2` . It encodes documents along 768 dimensions and it's one of the best performing models.

[1] We strongly recommend splitting large documents into sentences using a tokenizer such as `sent_tokenize` from `nltk` . This also has the advantage of naturally assigning multiple topics to a large document.

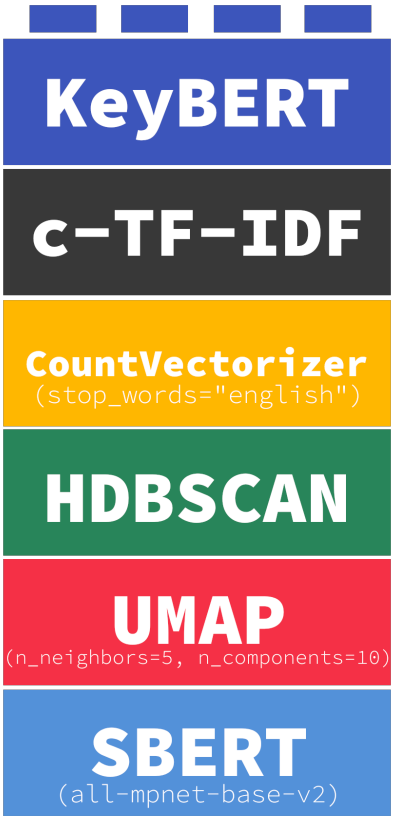

Figure 1.1: The pipeline of the BERTopic framework chosen for this project. (Image by the authors)

1.3.2 The dimensionality reduction phase

As we know, all machine learning algorithms suffer from the curse of dimensionality. Clustering algorithms are particularly sensitive to this, so it's good practice to apply dimensionality reduction methods, like *PCA*, *LDA*, *SVD* or *UMAP* prior to clustering analysis.

However, reducing the embedding space too much could damage the compositional structures that provide semantic distinction between different topics. **Therefore, there is a need for a balance between the quality of the embedding space and the efficiency of the clustering algorithm**.

We used the *UMAP* algorithm and empirically, we chose 10 output dimensions. *UMAP* preserves a lot of local structures even in lower-dimensional space, and thus works well with density-based clustering algorithms [Grootendorst, 2020].

1.3.3 The clustering phase

Now that the embedding dimensions have been reduced, documents are clustered by similarity. Density-based algorithms work very well at this stage because they don't force data points into clusters, but instead treat them as outliers.

HDBSCAN is a scale-invariant version of *DBSCAN*, which means it can handle clusters with different densities, as shown in figure 1.2, and does not require hyperparameter tuning when the points are scaled. It is the *BERTopic* default, so we decided to use it.

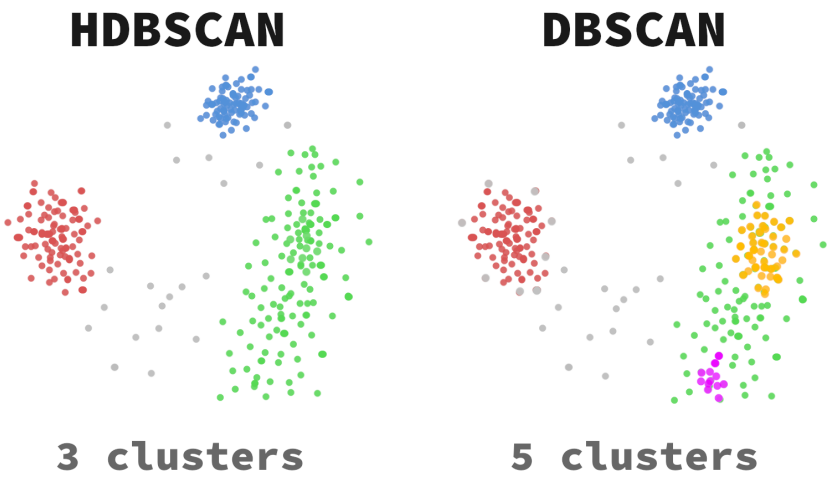

Figure 1.2: *HDBSCAN*, unlike *DBSCAN*, is able to detect clusters with different densities. On the left, *HDBSCAN* correctly finds 3 clusters and a set of outliers. (Image by the authors)

1.3.4 The topic extraction phase

Once the clusters have been determined, we need to interpret them. The set of documents contained in each cluster makes the model of a density-based clustering algorithm, but we can't understand what the topics are just by looking at the points, since **we ignore the meaning of the reduced embedding space**.

This phase is articulated in three steps: *tokenization of topics*, *weighting of tokens* and *keywords extraction*. Here we assume that the *Bag of Word* embedding is used for the 1st phase, while the *TF-IDF* is used for the 2nd phase and *KeyBERT* is used for the 3rd. Tokenize the clusters means to extract words, or n-grams[2] from their documents.

This way, the document-term matrix (DTM) is built.

> ✎ Note
>
> *LDA* builds the document-term matrix in the same way. The difference is that in *BERTopic* this is done after identifying the topics in order to interpret them, while in *LDA* it is done in order to find them.

Once the DTM has been extracted, *TF-IDF* is used to adjust the token frequencies to reflect the importance of tokens within a topic, relative to the overall corpus. Actually, *c-TF-IDF*, a variant of the classic *TF-IDF*, is used here. In essence, ***c-TF-IDF* is an application of *TF-IDF* to the union of all the documents in a cluster** — the *meta-document*. This makes sense, since we are looking for n-grams that distinguish a topic, not a document, from all other topics [3].

Finally, *KeyBERT* is used to ensure that the selected n-grams are not only relevant in TF-IDF terms, but also semantically meaningful with respect to the *meta-document*.

[2] The `n_gram_range` attribute of `BERTopic` class can be used to use n-grams instead of single words to define a topic. This can be very useful in terms of interpretability.

[3] *TF-IDF* gives weight to more frequent words than to less frequent ones, but excludes very common words on the assumption that they are most likely stop words, such as articles and prepositions.

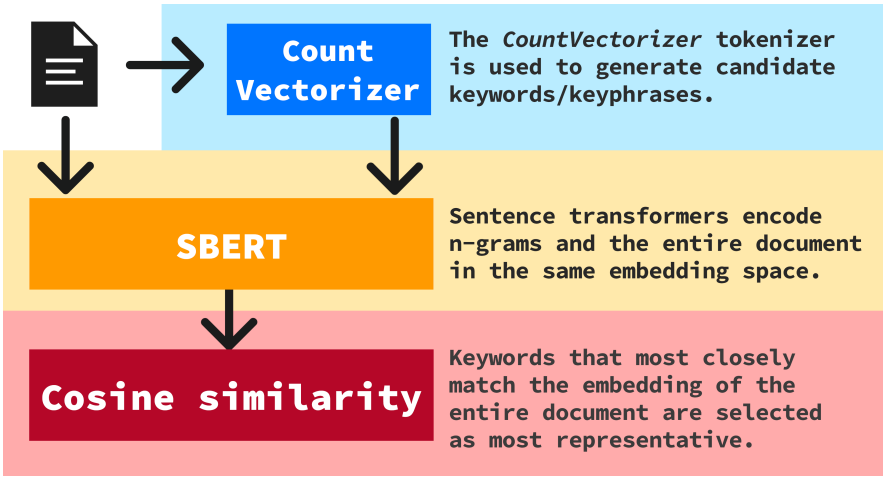

Figure 1.3: The three components of the *KeyBERT* model. (Image by the authors)

1.3.5 The post-processing phase

This is where the unsupervised algorithm ends. The user should now polish the results by *naming*, *reducing*, *merging*, and *visualizing* topics. BERTopic has several amazing tools for interpreting the results. We will use some of them in the following.

Part II

Analysis implementation

CHAPTER 2

Scraping and visualizing reviews

In this chapter, we will discuss web scraping reviews and some visualizations useful for exploring the data.500 reviews, extracted from the web pages of Amazon US, are plotted over time, displayed in word cloud diagrams, and their length is analyzed.

For our analysis, we chose the *HP OfficeJet Printer Thermal Inkjet* product, shown in figure 2.1, from *Amazon US*.

This printer has about 2,100 reviews that span the two years 2023 and 2024. Moreover, using the *Keepa* extension, We can see, as shown in figure 2.2, that the price is stable over time, so we can exclude that its variation systematically influences the ratings.

2.1 Scraping Amazon reviews

The `scraper.ipynb` notebook is used to scrape reviews of a product on Amazon.

Figure 2.1: The product whose reviews we are going to use for analysis. (Image from Amazon.com)

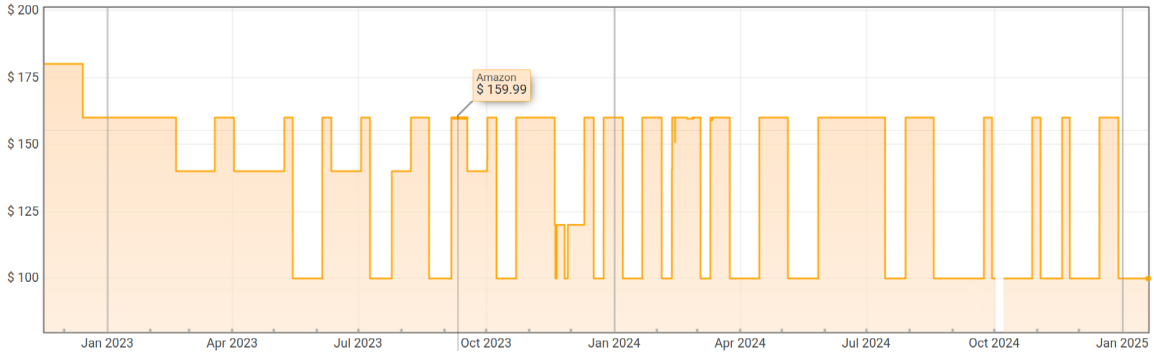

Figure 2.2: The history of the HP printer shows a rather stable price over the last two years. (Image by the authors)

> ⚠ Caution
>
> This code is intended for **educational and research purposes only**. The use of this code to scrape Amazon may violate their Terms of Service and could lead to legal consequences. By using this script, you agree that you understand these warnings and disclaimers.

2.1.1 Getting the URL

The `get_url` function returns the URL of a page containing 10 reviews of the given product with the given number of stars.

With a given filter configuration, Amazon limits the number of pages to 10. So I decided to filter by the number of stars. If available, this script will collect 500 reviews, 100 1-star reviews, 100 2-star reviews, and so on.

> ✎ Note
>
> This script was tested to work on January 16, 2025. As you know, web scraping depends on the website, which evolves over time. In the future, `get_url` may not return the correct URL.

With a given filter configuration, Amazon limits the number of pages to 10. So we decided to filter by the number of stars. If available, this script will collect 500 reviews, 100 1-star reviews, 100 2-star reviews, and so on.

> ➪ Important
>
> Amazon requires the user to be logged to view the reviews dedicated page. Therefore, you need to login with your browser and export your cookies, as a JSON file in the `/1-etl` directory. We used *Cookie-Editor* to do so.

2.1.2 Extract the text of the reviews

The `cleaner.ipynb` notebook is used downstream to the scraping algorithm to extract the text of the reviews from the `HTML` bodies. Using the handy *BeautifulSoup*

library, the `HTML` bodies are parsed and the 10 reviews are extracted from each page. For each review sample, we store the title, the content, and the stars.

> ✎ **Note**
>
> As previously noted, this operation is highly dependant from the Amazon HTML page code, which surely will change over time. Currently the pertinent fields are retrieved using the class of the parent of the `<span>` tag that contains the text.

2.2 Visualizing the ratings trend over time

The `ratings_over_time.ipynb` notebook is used to visualize the dataset of ratings by plotting them month by month and showing the distribution of stars, — the level of polarization of customers, over time.

From the monthly ratings plot, shown in figure 2.3, we can see that there are two peaks in the number of reviews in the months of January and December 2024. However, this doesn't necessarily mean that sales were particularly high during these periods, since the scraping process selected only the most relevant reviews, regardless of time uniformity.

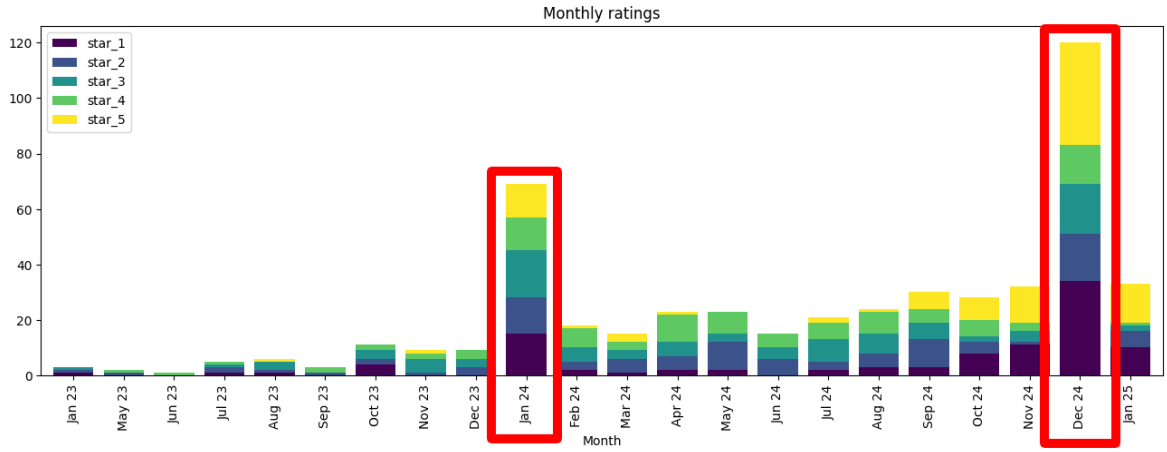

Figure 2.3: The number of monthly reviews.

When we stack the bars, as shown in figure 2.4 we see that the polarization of reviews increases over time. This means that customers are strongly disagreeing with each other.

2.3 Plotting Word Clouds

The `word_clouds.ipynb` notebook aims to show the most common words in both the titles and the content of the reviews. First, we analyze the 5-star reviews to identify the aspects that customers appreciate the most. Then the same process is repeated for the 1-star reviews.

This serves as a preamble to the Topic Modeling analysis that will follow. The most frequent words in the 5-star reviews, as shown in figure 2.5, are *easy, quality, price, new, in-time* and *office* The most frequent words in the 1-star reviews, as shown in figure 2.6, are *ink, cartridges, waste, scam, disconnects, paper, setup, subscription, wifi, support* and *app*.

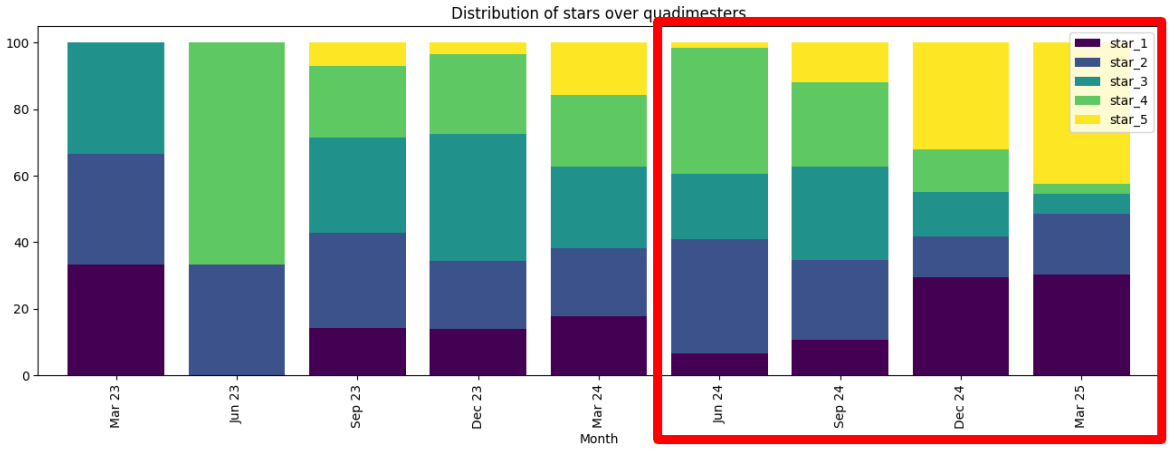

Figure 2.4: The distribution of stars in the reviews on a quarterly basis. The red rectangle in the figure shows the increasing polarization of customer opinions in recent years.

Figure 2.5: The word cloud of the 5 stars reviews.

2.4 Distribution of the lengths of the reviews

The `reviews_length_distribution.ipynb` notebook visualize the length of the reviews. The purpose is to determine how biased the dataset is toward a small set of customers.

As shown in figure 2.7, 14 reviews are shorter than 20 characters and thus were safely removed.

Figure 2.6: The word cloud of the 1 stars reviews.

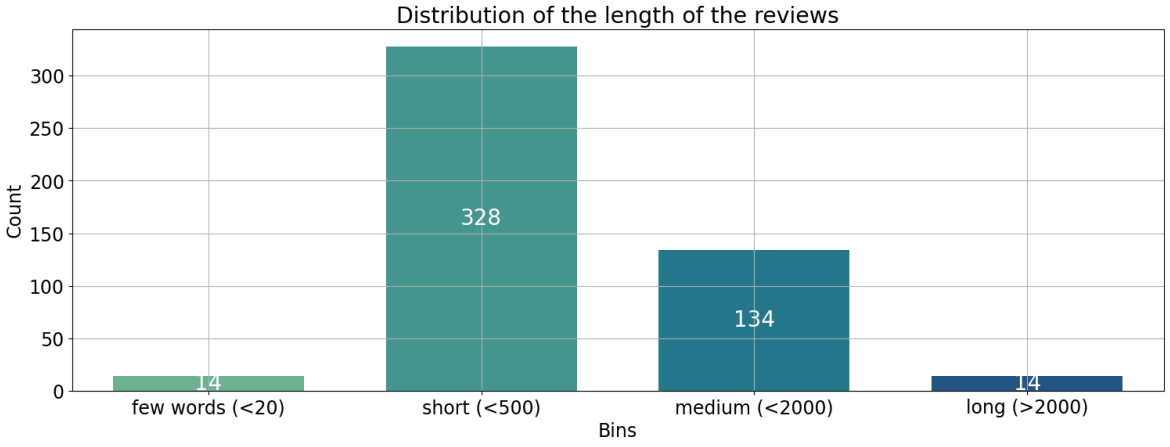

Figure 2.7: Looking at the distribution of review length, some reviews are very short and some are very long compared to the rest.

CHAPTER 3

Topic Modeling with BERTopic

After collecting the data, in this chapter we will tackle the topic modeling task by feeding it into the BERTopic model. In the following sections, we will cover both pre-processing and post-processing operations. Finally, we will discuss the extracted topics.

The `topic_modeling.ipynb` notebook performs the *Topic Modeling* task using *BERTopic*. The 500 collected reviews are clustered into semantically similar groups. From these clusters, topics are extracted and a probability distribution over the topics is calculated on the basis of the distances between the groups.

3.1 Data pre-processing

Before feeding the 500 collected reviews into *BERTopic*, we should consider a few issues.

3.1.1 Long 5-star reviews are sometimes paid

We could solve this problem by filtering out 5-star reviews that are longer than a certain number of words. However, since the goal of the analysis is to determine which aspects of a product drive overall customer perception, and since we selected the most influential reviews, we decided to ignore this fact.

3.1.2 Some reviews are very long

We solved this problem by splitting all the reviews into sentences. The original dataset of 500 reviews became a dataset of 2.821 sentences.

3.1.3 Some of the reviews are outliers

In the datasets there were a few amount of reviews that were removed. As shown in the previous figure, 14 reviews are shorter than 20 characters and were safely removed. In

addition, there was a Spanish review that caused the BERTopic model to define a cluster just for that review, since an English-only tokenizer was chosen.

The obtained model is `model.bertopic`. It is the result of several attempts with multiple hyperparameter configurations. As the documentation states, BERTs is stochastic, so running the code multiple times may produce different results[1].

3.2 Post-processing operations

BERTopic extracted 54 topics plus one "outlier" topic. The latter is conventionally assigned id -1 and is the result of all sentences that *HDBSCAN* could not classify as either core or boundary.

3.2.1 Merging similar topics

Some of the 54 topics can be merged because they are semantically very close sentences. The merging decision can be seen in `raw_topics.txt`.

The `BERTopic`'s `merge_topics` method was thus called accordingly. This method also requires the training corpus, because once two topics have been merged, the *c-TF-IDF* score must be recomputed because the *meta-document* has changed.

This results in the 54 topics being reduced to 40. The post-processed model is `model_postproce`
.

3.2.2 Naming the topics

For the sake of clarity, the remaining 40 topics have been assigned custom labels. The interpretation was on the basis of the extracted n-grams and the most relevant sentences in the cluster. They are shown in figure 3.1.

3.3 Understanding the results

The final result is shown in figure 3.2.

For visualization porpuses, the 10 UMAP dimensions are further collapsed into 2 axes. This results in some of the topics overlapping. However, we can clearly see that semantically distant topics reside on distant islands.

The matrix in figure 3.3, shows the correlation between each pair of topics. The distance between topics is not uniform. For example, the topics "ink and cartridge" and "low cartridge" have a correlation value greater than 0.8, as one might expect.

3.3.1 Calculating the topic distribution

If we select a particular sentece, we can display the probability distribution over the topics. Consider the following document.

[1] It is highly recommended to use the GPU if available. Both the embedding phase with the SBERT model and the fitting of the BERTopic were dramatically faster with GPU support.

Figure 3.1: The 40 extracted, post-processed topics.

➪ **The interactive version can be accessed here.**

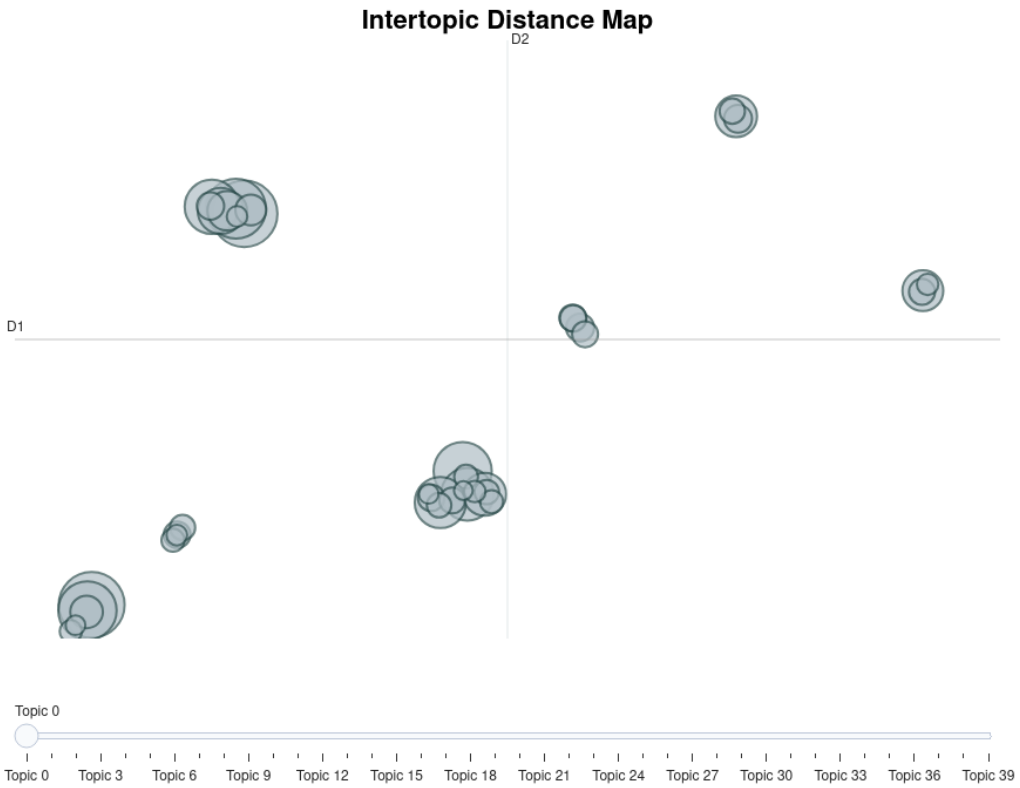

Figure 3.2: The map of the 40 topics on a reduced 2-dimensional embedding space. The size of the circles is proportional to the frequency on that topic.
➪ **The interactive version can be accessed here.**

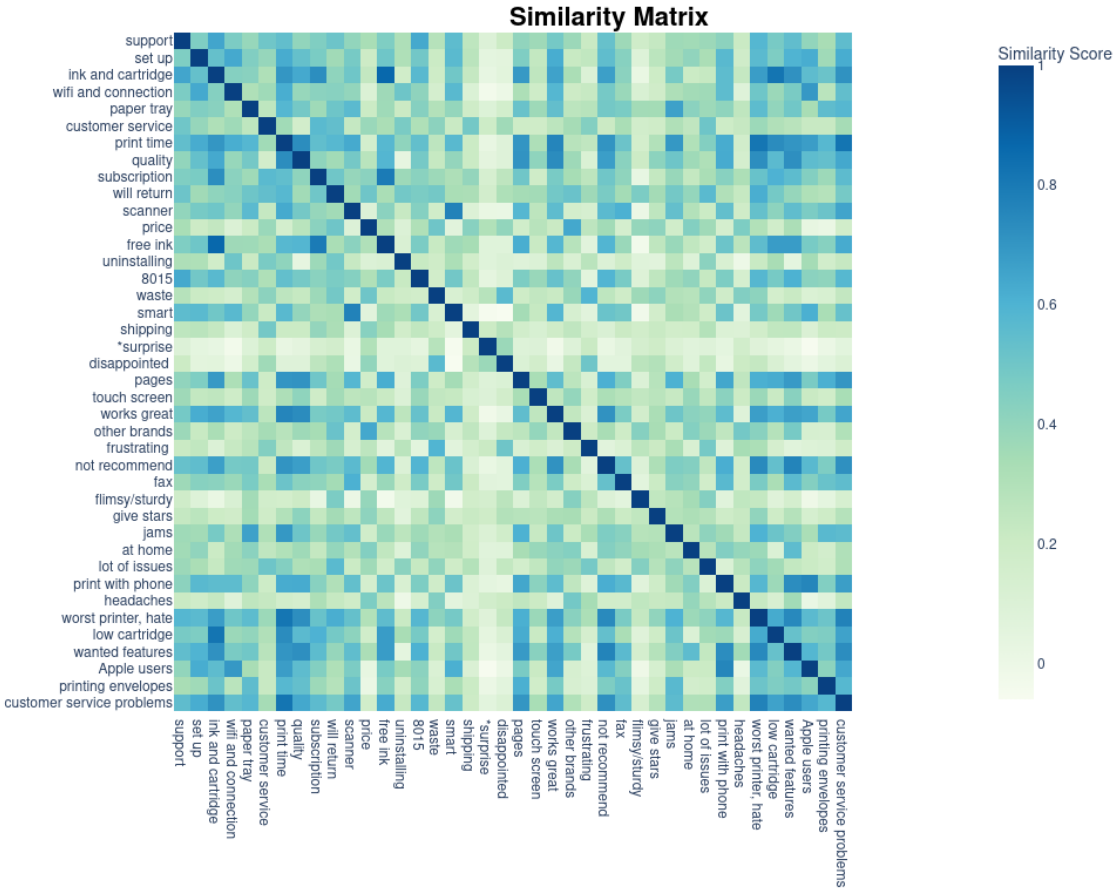

Figure 3.3: The similarity matrix of the topics shows how much the topics are related — how close they are in the reduced embedding space.
➪ **The interactive version can be accessed here.**

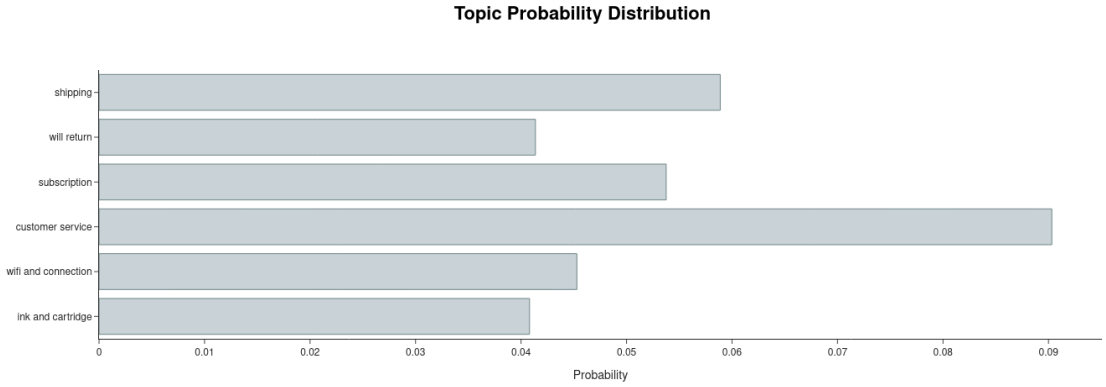

Figure 3.4: The most likely topics of Jerry's comment.
➪ **The interactive version can be accessed here.**

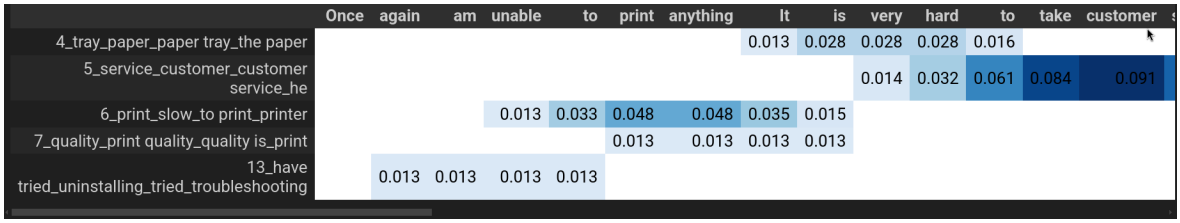

Figure 3.5: The approximated distribution over the Jerry's sentence.
➪ **The interactive version can be accessed here.**

> Jerry, January 28, 2024, extract from "*Terrible, Terrible, Terrible*".
> Once again I am unable to print anything. It is very hard to take customer service reps and their supervisors seriously when they contradict themselves don't listen to you and don't execute on their promises. In general, I think they have a serious attitude problem in customer service.

We can see that Jerry has had a difficult time with HP customer service. So it come to no surprise that the topic with greater confidence is "customer service", as shown in figure 3.4. However, one might wonder which part of the sentence contributed the most to determining the probabilities of the topics.

The `approximate_distribution` method uses a sliding window to estimate the similiarity of each token, and its neighbourood, with each topic. The output of this method for Jerry's review is shown in figure 3.5.

3.3.2 Visualizing the topics

The two-dimensional map in figure 3.6 shows the final clustering analysis. In the interactive version, we can move the mouse over the points to see the reviews related to a topic. The BERTopic model did a great job. As shown in figure 3.7, All comments about a particular aspect of the printer, such as problems with jamming, the ability to print with the phone, and the robustness of the printer, are collected together.

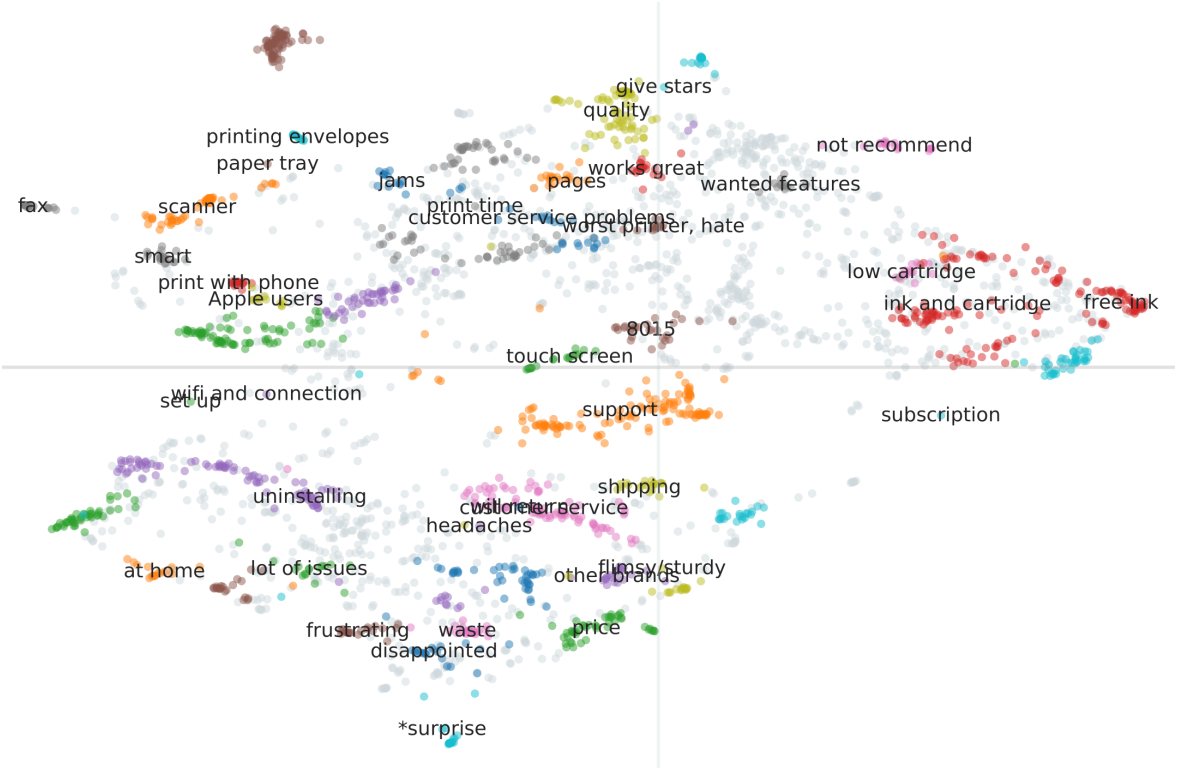

Figure 3.6: The map of all the 2.821 extracted from the 500 reviews. Each island contains customers' opinions about a particular aspect of the printer.
➪ **The interactive version can be accessed here.**

3.3.3 Why is this useful?

This last interactive chart can save a lot of time for managers who want to improve the product. There's no need to go through all the reviews one by one; they're collected and organized according to their importance.

The manager should just decide which topic seems more interesting to him. If they want to know if HP's customer service is doing its job well enough, they can take a look at the items within the respective cluster.

The sentences more or less agree on the unhelpful experience they've had with the help.

As for the mobile app, customers are largely satisfied, although someone is annoyed by the frequent login requests.

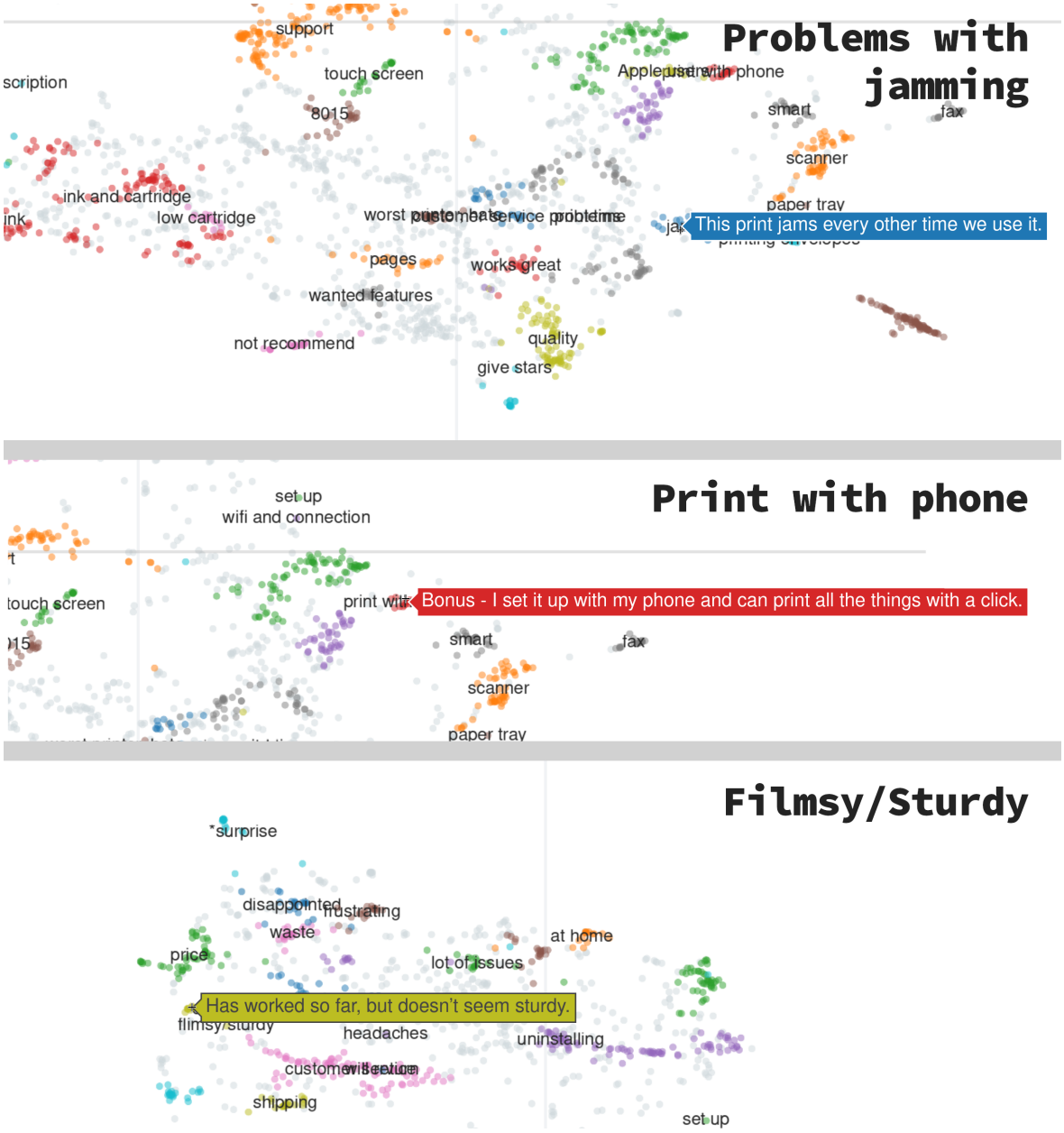

Figure 3.7: Three samples extracted from different topic in the map.
➪ **The interactive version can be accessed here.**

CHAPTER 4

Sentiment analysis with TextBlob

In this chapter, we conclude the analysis by enriching the topics extracted above with a sentiment analysis.

The `sentiment_analysis.ipynb` notebook organizes the previously found topics in a polarity x subjectivity matrix. To obtain these values, the representative documents in each topic are analyzed and the results are averaged.

4.1 TextBlob for sentiment analysis

TextBlob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing [Loria, 2024].

TextBlob performs sentiment analysis using a lexicon-based approach[1]. It utilizes prebuilt sentiment lexicons, such as the *Pattern* library, to analyze text.

4.2 Calculation of the topic sentiments

To determine the sentiment of the topics, we computed the average *polarity* and *subjectivity* over all the most representative docs that BERTopic has extracted from each cluster in the `get_topic_info` method.

4.3 Visualizing the sentiments

The results of the sentiment analysis are shown in figure 4.1. The information shown in the chart can be used togheter with what has been said in section 3.3.3 to identify the most critical aspects of the product.

[1]The library uses a predefined dictionary of words and their associated sentiment scores — polarity and subjectivity. It evaluates the sentiment of individual words in the text and combines these scores to determine the overall sentiment of the text.

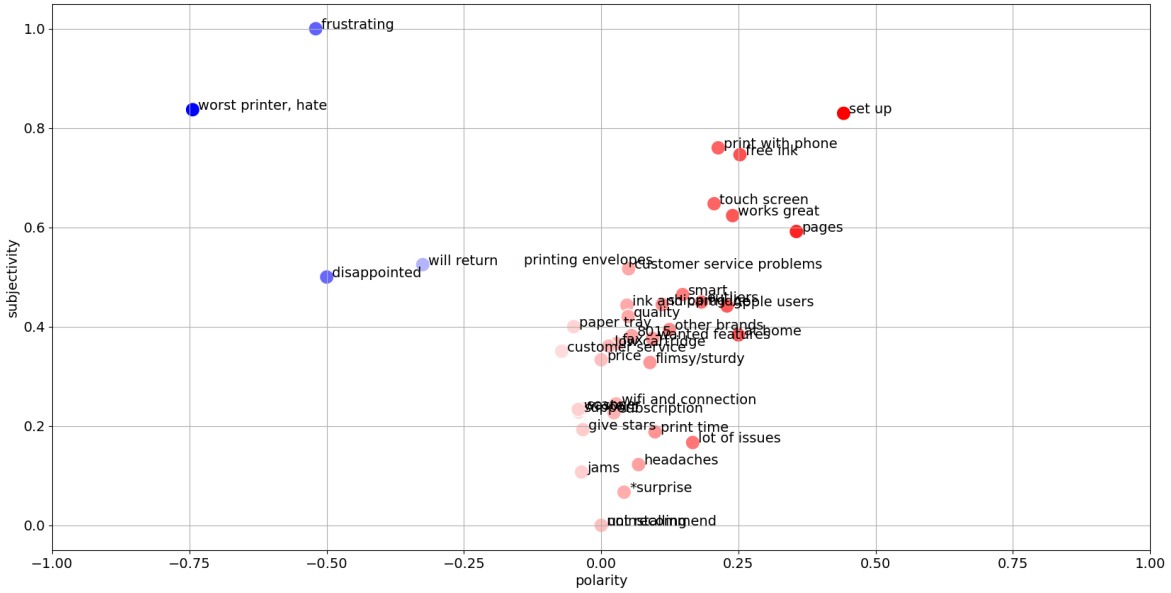

Figure 4.1: The topics are plotted in a polarity x subjectivity matrix. The topics farthest to the left are the most negative, while the topics farthest to the right have more positive than negative reviews.

Once again, we see that the customer service issue receives a slightly negative average polarity, while the ability to print with the phone is generally appreciated, with an average polarity of almost 0.25.

Bibliography

ANGELOV, D. (2020), «Top2Vec: Distributed Representations of Topics», URL https://arxiv.org/abs/2008.09470. (Cited at page 3)

GROOTENDORST, M. (2020), «Topic Modeling with BERT», https://towardsdatascience.com/topic-modeling-with-bert-779f7db187e6. (Cited at page 4)

GROOTENDORST, M. (2022), «BERTopic: Neural topic modeling with a class-based TF-IDF procedure», URL https://arxiv.org/abs/2203.05794. (Cited at page 3)

LORIA, S. (2024), «TextBlob: Simplified Text Processing», https://textblob.readthedocs.io/en/dev/. (Cited at page 21)

MOODY, C. E. (2016), «Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec», URL https://arxiv.org/abs/1605.02019. (Cited at page 2)

SHERIDAN, S. (2024), «What Is Topic Modeling? A Beginner's Guide», https://levity.ai/blog/what-is-topic-modeling. (Cited at page 2)