A Summarization Pipeline For Album Reviews

Abhishek Dhar

Abstract

Album reviews are a valuable source of information for music consumers. However, they can be long and tedious to read. This project introduces a pipeline that summarizes album reviews through extractive and abstractive summarization techniques. The proposed pipeline has several advantages over existing methods for displaying album reviews, including its ability to capture the most important aspects of the reviews, and its potential to be used as a content moderation tool by ignoring extreme or unrelated reviews. The pipeline involves a series of NLP, machine learning, and graph related techniques to generate a summary. It uses parts of speech tagging to focus on the more important types of words, clustering and transformers to group similar documents together and generate summaries from extracted sentences respectively, and a network of document words to implement the position rank algorithm. The code for this project can be found here: https://github.com/Xenox473/review_summarizer/tree/main.

Introduction

Reviews are often in the form of short opinions formed by various authors that grow in number along with the popularity of the object being reviewed. A reader might be interested in reading these reviews to better understand the overall sentiment of the product. However, it's easy to get overwhelmed by the ever increasing number of reviews. This makes it difficult for the reader to filter out the most important reviews and instead, they must rely on other tools to surface the most appropriate reviews for them. Some of the most popular ways of surfacing reviews is either by displaying the reviews chronologically with the latest reviews being shown first or by popularity, which requires other readers to vote for the reviews they think are most helpful. While these two approaches are often useful, they also are incapable of handling a large collection of reviews. For example, showing the newest reviews first forces the reader to disregard the older reviews due to a lack of time, or relying on community votes may cause the voters to fall prey to the "Hive Mentality" where they end up voting for, or against, a review based on its present vote count. This self-serving process retains the reviews deemed worthy by an initial set of voters and disregards any other review that came in too late to develop any traction.

The purpose of this project is to explore summarization techniques in order to generate a single review from the collection of reviews, such that the generated reviews presents the most relevant information to the reader. Some applications in which this would be useful is:

1. It allows the reader to gain a holistic summary of the product without having to peruse all the reviews.

2. It can be used as a tool for automatic content moderation by only displaying reviews that represent the majority sentiment and filter out extreme/spam reviews.

There are two kinds of summarization techniques: extractive and abstractive. Extractive summaries work by returning sentences deemed the most important in a corpus. For example, a popular Reddit summarization bot worked by assigning each word in an article a TF-IDF score and summing up these scores for all the words in a sentence. Finally, the output is the top-N sentences according to their total scores. Meanwhile, abstractive summaries work using transformers or other large language models to generate new sentences using the corpora. Due to the nature of these models, they're often restricted to a certain window size - which is much smaller than the number of tokens that belong to all reviews of an object. This makes them unusable for generating a holistic summary of all the reviews.

As a result, the plan for this project is to use a combination of the two summarization types to extract the most important sentences for a collection of reviews and generate a new summary using a transformer model.

Prior Works

There have been several published papers that have influenced the decisions made in this project. This section covers them.

TextRank (2004)

This paper introduced a novel way to summarize text by approaching the problem through the lens of network science. The authors suggest that given a graph of any document, a node represents a word and an edge between two nodes represents the number of times those two words co-occur within a window. Once this graph is constructed, the PageRank algorithm can be used to rank the nodes/words by order of importance. This ranking makes it possible to extract key phrases from the documents.

ColabRank (2008)

This paper built off of the idea in TextRank by introducing the idea that instead of processing documents independently, it might be better to generate a graph of words using the target document along with other documents that share the same context.

PositionRank (2017)

This paper once again builds upon the previous two by encoding the position of a word in a document into the TextRank's algorithm such that words that occur earlier and more frequently in a document are ranked higher in importance. This new approach produced the best results, beating the paradigms it was built on and TF-IDF as well.

Methodology

Dataset

The data for this project was sourced from https://1001albumsgenerator.com/stats. The reviews and their scores for each album were scraped using a custom python script. For the purposes of this report, all examples refer to the corpus of Abbey Road reviews. The Abbey Road corpus consisted of 1002 mostly English reviews.

Preprocessing

In order to use this data some basic preprocessing of the reviews needed to be done. This included the following:

1. Tokenization: All words were converted into their respective tokens.
2. Stop words were filtered out.
3. Words that weren't nouns or adjectives were filtered out. According to the PositionRank paper, this provided the best results.
4. Lemmatization: All the preprocessed tokens are used in their lemma form.
5. Vectorization: Vectors for each review are calculated by averaging the vectors of all its tokens. Words or reviews without any vectors are filtered out. These are usually non English words.

Process

Figure 1: Illustration of the steps taken

Figure 1 shows the steps determined necessary to generate a review for an album's corpus. Preprocessing comes first using the steps described earlier. Following that, we treat each review as a singular document and cluster them using the agglomerative clustering method. A hierarchical approach was used since we don't really know the optimal number of clusters to divide the corpus into. By using this method, we're able to have the corpus naturally split into several clusters that contain documents that are similar according to the cosine similarity of their vectors.

We are now left with hundreds of clusters, most of which only contain one document. These are the most extreme of reviews and shouldn't be accounted for. Thus, we filter down the clusters to those that have a number of documents greater than the average count of documents across all clusters. This significantly reduces the number of clusters to work with. We also filter the clusters based on a cluster's variance in scores. We only keep the cluster's with variance within the 25th percentile. This ensures that the clusters consist of similar reviews.

We can now extract the key sentences for each cluster. This is done using a customized version of the PositionRank algorithm. For each cluster we create a graph of words with the edges representing how often two words co-occur within a window of size 4. Once that is constructed, we calculate each word's positional rank. This is done by

summing up the inverse of a word's positions within a document, and averaging each of the sums across documents within a cluster. These scores are then used in the Networkx's implementation of the PageRank algorithm as the "personalization" parameter. This parameter influences the bias in the algorithm by adjusting the preference of each node according to their positional scores. We are now left with a graph that consists of nodes that have been scored by the algorithm. This graph can be used to either find the key phrases or the most important sentences in a cluster.

Key phrases are extracted by calculating the scores for all (one, bi, or tri)gram phrases that match the following regex patterns described in the PositionRank paper: (adjective)*(noun). The most important sentences are determined by the total of all of a sentence's words' score. We return the top 3 key phrases/sentences per cluster.

This collection of key phrases or sentences can now be passed into a large language model to generate a summary of all the reviews. For this project, the distilbart and GPT 3.5 Turbo models were used. These models were able to merge the most important sentences into a more cohesive synopsis of all the reviews.

Results

This section steps through an example of the process while showing relevant results. As mentioned earlier, the dataset for the Abbey Road corpus consisted of 1002 reviews. The following are examples of reviews:

1. This album is testament to the genius of Sir George Martin. You know how when folks in the UK want to honor someone the knight them and call them "sir"? In the USA we should honor them by adding the middle name F-ing. This guy right here would be Sir George F-ing Martin.
2. This is, no hyperbole, the greatest album of all time. It's the last one the Beatles made all together, and they almost tore each other apart doing so. There are tracks where Lennon doesn't appear, tracks where McCartney doesn't feature as prominently as he would have previously, and tracks where you can almost feel the frustration. In spite of it all, what they put together was incredible and, as always, greater than the sum of its parts. Here Comes the Sun is George Harrison's best work. I Want You (She's So Heavy) is a typical complex contribution from John. Paul took "some crap John wrote in India" and

```
weaved it together with some of his own material to make the masterpiece that
is the Abbey Road medley. Ringo...well, Octopus's Garden is pretty good, as far
as Ringo's stuff goes. He did finally get a solo, which is nice. The album is
so packed that I haven't even mentioned several other greats: Come Together,
Something, and Because are all-timers. Best track: the medley
```

The first review doesn't contribute much to understanding the quality of the album while the second review goes into great detail. This wide and varied range really makes it difficult for a reader to parse through and distinguish the most important reviews.

review	score	processed_review	vector
This album is testament to the genius of Sir G...	5.0	[album, testament, genius, sir, george, martin...	[-0.2669848, 0.024653222, 0.047056668, -0.2378...
This is, no hyperbole, the greatest album of a...	5.0	[hyperbole, greatest, album, time, beatles, tr...	[-0.020217512, 0.14377919, 0.07309127, -0.0901...
The fact that this album has a Wikipedia page ...	5.0	[fact, album, wikipedia, page, song, important...	[-0.14276828, 0.236743, -0.07973867, -0.019563...
That weight their taling about is john legons ...	4.0	[weight, taling, john, legons, fat, nutslol]	[-0.29676384, 0.37898502, -0.020394502, 0.0915...

Table 1: Review data after preprocessing

Once the data is processed, it is fed through the hierarchical clustering algorithm. This process breaks down the collection of documents into 221 clusters. Filtering down the clusters based on the process described earlier narrows down our dataset to 5 clusters.

	score							
	count	mean	std	min	25%	50%	75%	max
cluster								
0	13.0	3.769231	0.832050	3.0	3.0	4.0	4.0	5.0
1	4.0	5.000000	0.000000	5.0	5.0	5.0	5.0	5.0
2	7.0	3.857143	1.214986	2.0	3.0	4.0	5.0	5.0
3	5.0	4.200000	1.095445	3.0	3.0	5.0	5.0	5.0
4	2.0	4.000000	1.414214	3.0	3.5	4.0	4.5	5.0
...	...	...	...	...	...	...	...	...
216	1.0	5.000000	NaN	5.0	5.0	5.0	5.0	5.0
217	1.0	5.000000	NaN	5.0	5.0	5.0	5.0	5.0
218	1.0	5.000000	NaN	5.0	5.0	5.0	5.0	5.0
219	1.0	5.000000	NaN	5.0	5.0	5.0	5.0	5.0
220	1.0	4.000000	NaN	4.0	4.0	4.0	4.0	4.0

Table 2: Summary Statistics of the Clusters

For each cluster, we can now extract the top 3 key phrases and sentences for each cluster. The following provides examples of some of the key phrases and sentences extracted from the clusters.

1. `[('fucking abbey road', 0.6087258994696664), ('abbey freaking road', 0.6003704136129285), ('abbey road', 0.5709934552480703), ('abbey', 0.2928910187712208), ('road', 0.2781024364768495)]`
2. `[('good', 0.6716559858162368), ('time', 0.22555264409441522), ('.classic', 0.10279137008934784)]`
3. `[('masterpiece masterpiece', 0.9691566348281901), ('masterpiece', 0.48457831741409507), ('technicolor genius', 0.28783639982322173), ('top shelf', 0.12787898612256068), ('songwriter', 0.09970629664012233)]`
4. `[('star album', 0.3836189701214502), ('amazing album', 0.25851668903824915), ('album', 0.19685488386324693), ('star', 0.18676408625820323), ('hearing professional', 0.0926162230004052)]`
5. `[('entire album', 0.12884617868309212), ('favorite album', 0.1190643165942657), ('favourite album', 0.11236941666459914), ('perfect album', 0.10630437108039101), ('classic start', 0.08845181758479567)]]`

1. 'It’s abbey road',
2. '10/5….classic..one of the best of all time',
3. 'A masterpiece',
4. 'This just might be a 5 star album...',
5. "I can't still remember the first time I heard the entire album."

Once we've extracted our sentences across the clusters, we can pass them into our large language models. This text should now be able to fit within the model's token window size. Below are results generated by distilbart and GPT 3.5 turbo respectively along with a snapshot of some of the sentences passed into each model.

1. Abstractive Summary generated by distillbart model:

   It is impossible not to give five stars for a album like this.. its abbey road. A masterpiece. The best. It's probably my favourite album of all time.. the easiest 5 i'm going to give this entire list, this is perhaps the perfect album . This just might be a 5 star album.... I can't still remember the first time I heard the entire album .

2. Abstractive Summary generated by GPT-3.5:

   This album, Abbey Road, is truly a masterpiece and the best of its kind. It's impossible not to give it five stars. Personally, it's my favorite album of all time and I have no doubt that it deserves a perfect rating. I still vividly remember the first time I listened to the entire album. There's not much more to say about this album that hasn't already been said; it's a classic and one of the best of all time. If this isn't a masterpiece, I don't know what is. Abbey Road is truly exceptional.

```
['A masterpiece',
 'The best',
 'It is impossible not to give five stars for a
album like this.',
 'its abbey road',
 "It's probably my favourite album of all time.",
 'the easiest 5 i'm going to give this entire list,
this is perhaps the perfect album.',
 "it's abbey road.",
 'This just might be a 5 star album...',
 "I can't still remember the first time I heard the
entire album.",
 "Not much I can say about this album that hasn't
already been said.",
 '10/5….classic..one of the best of all time',
 "If that's not a masterpiece, I don't know what
is.",
 'It's abbey road']
```

These summaries seem pretty on point with the overall sentiment across reviews. Abbey Road is a classic, after all.

Conclusion

Through this project we've been able to build a pipeline that can summarize a collection of album reviews using a combination of natural language processing and network science tools. This pipeline should help improve the user experience for any future reader by providing succinct summaries of the majority opinion for an album and ignoring redundant or non relevant reviews.

There is room for customization and improvement. The parameters can be tweaked to have longer summaries or cover a wider range of opinions. The prompt used to instantiate the GPT model could be further engineered to provide a more concise summary. Due to the specificity of this project's use case and the vastness of the dataset, it was difficult to build a robust testing procedure. One idea not yet implemented is to perform some sort of A/B testing such that a tester is provided two summaries - one generated from our pipeline, and another containing a random selection of sentences from the corpus - and is asked to pick the most representative summary. The results from this experiment could help finetune the various facets of the pipeline.

Finally, a limitation to this summarization technique is that it doesn't do semantic summarization, which could be beneficial when working with an opinion-based corpus such as this one. A next step would be to build in the semantic understanding within this pipeline through more detailed parts of speech tagging. Another step that could improve this project is the ability to summarize other languages. While this corpus is primarily English, there are other languages as well that could enhance the context of the generated summaries if they were understood instead of being ignored.

References

1. TextRank: Bringing Order into Texts
2. CollabRank: Towards a Collaborative Approach to Single-Document Keyphrase Extraction
3. PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents
4. TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction
5. Exploring Different Keyword Extractors — Graph Based Approaches | by Ishan Shrivastava | GumGum Tech Blog | Medium
6. SMMRY, the Algorithm behind Reddit's TLDR Bot | by Matthew Plaut | Medium
7. sshleifer/distilbart-cnn-12-6 · Hugging Face
8. API Reference - OpenAI API