Recognizing themes in Amazon reviews through Unsupervised Multi-Document Summarization

Hanoz Bhathena
hvb2108@stanford.edu

Jeff Chen
jc1@stanford.edu

William Locke
wlocke@stanford.edu

Abstract

Amazon now has an overwhelming number for reviews for popular products, and shoppers are routinely spending a lot of time reading through hundreds or thousands of reviews while making mental notes of important themes. We propose a system for unsupervised extractive summarization using deep learning feature extractors combined with several different models: k-means, affinity propagation, DBSCAN, and PageRank. We also propose a system for unsupervised abstractive summarization using a deep learning model. These summarization models capture the major themes in all the reviews of a product and optionally report the number of users who mentioned the same theme, hence giving shoppers a sense of salience. Since summarization in an unsupervised learning is notoriously difficult to evaluate, we propose a suite of metrics: ROUGE-1, semantic similarity, sentiment accuracy, attribute match, and content preservation, as ways to measure the quality of the summary. We found that extractive summarization is able to capture important themes, though sometimes the exemplar sentence is not well chosen. We found that abstractive summarization is able to generate human-readable text, though content preservation is challenging. Our baseline of k-means clustering scored a 3.48 and 3.44 for attribute match and content preservation, respectively and we achieved 3.72 and 3.93, respectively, with PageRank.

1 Introduction

The number of reviews for products on Amazon is overwhelming as it's common to see products with more than 1,000 reviews. The shopper who just wants to know the common themes is left to read pages upon pages of reviews and take mental notes of recurring topics. We would like to have a system that outputs themes when given a set of product reviews on Amazon. Since labeled datasets from Amazon reviews to summaries do not exist and are very costly to produce, we chose to take on the problem of generating summaries using unsupervised learning with no ground truth summaries available to us.

We first propose an unsupervised extractive summarization algorithm using sentence embeddings along with clustering to automatically extract the most common themes among a set of reviews. Concretely, given a set of reviews for a particular product, our system will output a list of sentences with the most salient themes in the review set. We also propose an unsupervised abstractive summarization model that will directly write a summary when given a subset of reviews for the product.

We face several distinct challenges in achieving this. First, since the number of salient themes found in a given set of reviews might differ between products, our system will need a way of deciding upon the appropriate number of themes to assign for to a given product. Second, the summarization should express a given theme in a way that is concise yet comprehensive. Third, since our problem is unsupervised, we need to establish an appropriate set of metrics that can act as proxies in place of ground truth comparisons.

2 Related Work

A recent paper demonstrates the use of centroid-based summarization through word embeddings [1]. Rossiello et al. determines the central topic of an article by summing all the important words as dictated by term frequency-inverse document frequency (TFIDF), and then encodes sentences by aggregating the word vectors of each word. And finally calculates the distance by the cosine distance. This method performed better than centroid-based summarization using a Bag of Words model [2], where vectors for sentences are determined by their word-frequencies. *Analysis of Adjective-Noun Word Pair Extraction Methods for Online Review Summarization* by Yatani et al. demonstrates that showing the importance of summaries is useful for users to build an impression about the product [3]. ROUGE-N score has been used as a way to evaluate summarization algorithms; Chu et al. [4] demonstrate that it can be used as a proxy in unsupervised settings as well. They also demonstrate the possibility of using a trained text classifier as an indicator of the accuracy of the summary. This paper describes a way to automatically generate abstractive summaries in an unsupervised learning setting by making use of sequence to sequence autoencoders and sequence similarity. We discuss more of the details behind this model including our implementation in the Abstractive Summarization sections of this paper. Finally, there has been more work on abstractive summarization using neural nets [5].

3 Dataset

Our dataset is sampled from a large dataset consisting of 142.8 million product reviews (along with metadata) collected from Amazon for the period May 1996 to July 2014 [6]. This larger dataset is divided into 24 product categories (e.g., books, electronics, video games, etc.). We base our work on a subset of reviews made for products in the electronics category that contains around 7.8 million reviews for 476k products. Each item in this set of reviews also includes a number of meta-attributes such as a reviewer name and ID, an Amazon Standard Identification Number (ASIN), the ratio of users that marked the review helpful to those that found it unhelpful, the overall star rating (out of 5.0), and the date and time the review was made. 90% of the reviews have less than 200 words as seen in Figure 1. Additionally, 20% of (electronics) products have 10 or more reviews. An example of a review is shown in Figure 2.

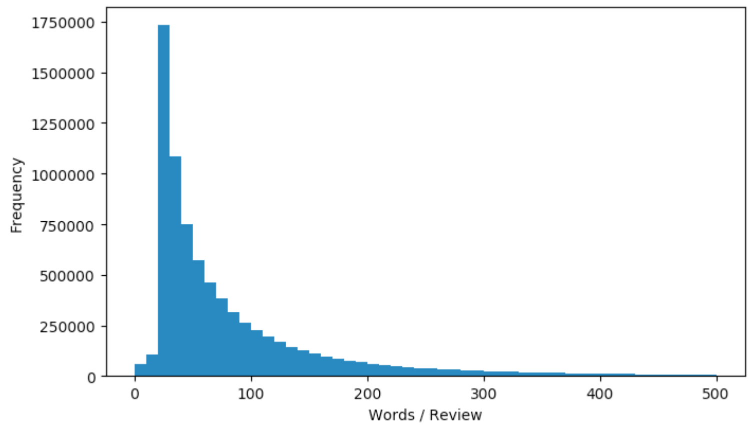
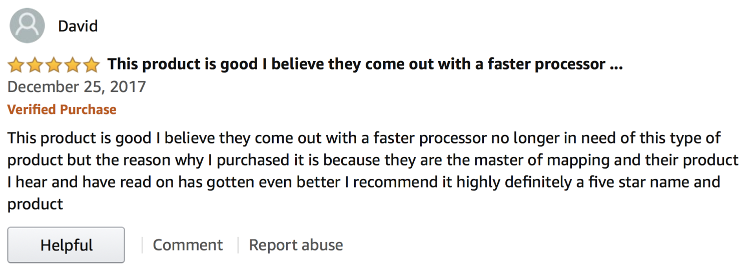


Figure 1: Histogram of number of words per review.

4 Methods: Models and Evaluation

4.1 Machine Evaluation

Currently, the most widely used metric to evaluate summarization is the ROUGE-N score, which measures the overlap of n-grams between the reference summaries and those generated by an au-


David

This product is good I believe they come out with a faster processor ...

December 25, 2017

Verified Purchase

This product is good I believe they come out with a faster processor no longer in need of this type of product but the reason why I purchased it is because they are the master of mapping and their product I hear and have read on has gotten even better I recommend it highly definitely a five star name and product

Figure 2: Sample review

tomatic summarization system. However, ROUGE does have its limitations when it comes to correlation with human judgment of the quality of a summary since having overlapping words does not necessarily indicate the meanings are the same. Furthermore, ROUGE becomes a particularly tricky metric to evaluate unsupervised text summarization since we do not have *real* ground truth summaries. Therefore, we generally adopt heuristics like taking the ground truth as the entire review set that needs to be summarized.

We therefore need to come up with additional/alternative metrics and we choose two such metrics to evaluate our unsupervised summarizations in an automated way.

Review sentiment classifier: A review generally has a rating between 1 to 5. Using a trained review rating classifier we can score the documents we are trying to summarize and the summaries we generate using this classifier. If the difference between the average ratings classification for the summaries and the raw product reviews is small we have a good summarizer. This metric inherently measures the consistency of sentiment or tonality between the original reviews and their summaries which should be as high as possible.

Semantic similarity: In addition to word overlap and sentiment consistency, we need an automatic measure of the overall content preservation. We discuss this in more detail in the human evaluation section below, but replicating the human process for content preservation measurement is a hard problem for a machine. We therefore rely on semantic similarity as a proxy for this. After encoding every sentence into a fixed dimensional vector representation we calculate its cosine similarity with the original reviews. The sentence embeddings, and by extension the word embeddings which are the inputs to the model that creates them, will be more similar if the summaries contain words which are similar meaning to those in the original reviews. However, these words do not need to be exact matches like in the case of ROUGE. For example, $customer$ and $client$ would have a high semantic score but a zero ROUGE score.

4.2 Human Evaluation

While automatic evaluation metrics such as ROUGE, semantic similarity, and sentiment accuracy provided a good indication of the performance of our models, we found it helpful to include a number of human evaluation metrics to either corroborate or provide additional insight into the performance of each method. To this end, in our clustering-based approaches, we primarily used human evaluation to identify how well exemplar sentence summaries preserved the theme content of their respective clusters as well as an attribute match evaluation metric that determined how reflective the sentiment applied to a given feature is of the general sentiment applied to that same feature in the cluster. We applied human evaluation by taking a random sample of summary sentences and scoring them using a standard Likert scale (between 1 and 5). For content preservation, we asked the question: "Does the content of the summary represent the most commonly described features in the cluster review sentences?" For attribute match we asked the question: "Does the summary represent how this feature is generally described in the cluster review sentences?" We applied the same two human evaluations to abstractive summarization and PageRank, but instead of comparing the summary to a review cluster we compared it to the entire review set.

4.3 Sentence Embeddings

Before we perform any summarization, we need to first encode review sentences using a sentence encoder. We experimented using the pre-trained text encoders available through Tensorflow Hub [7]. We utilized two types of text encoders: neural network language model [7] based on Bengio et al. [8] and the Word2Vec model by Mikolov et al. [9]. The neural network language model is an n-gram language model where a window of the previous n-1 words is used to predict the current word. The Tensorflow Hub NNLM encoder was trained on the English Google News 200B corpus [8]. The sentence embeddings are 128-dimensional. Word2Vec has become a popular way to encode words using a dense, fixed size vector representation and has spurred great innovations in NLU over the last few years. The sentence embedding we generate here is simply the weighted average of the word vectors of all the words in the sentence and is 500-dimensional.

4.4 Extractive Summarization Algorithms

After finding sentence embeddings for each sentence in a particular product review set, we use these embeddings as features that go into different models described below and generate a summary. We aim to select sentences which together will cover both the most salient themes and also give maximum coverage on non-overlapping concepts. For clustering algorithms, we cluster sentences using their dense vector representations and systematically pick sentences from each cluster to make up our end summary. For ranking algorithms, we pick sentences based on their rank.

4.4.1 K-means Clustering

Using the sentence embeddings as features, a k-means algorithm was run on the sentences. We fixed the number of clusters to 5. We chose this cluster count using the number of possible ratings as a guide. Note that the ideal method would be to pick clusters that have a one to one correspondence to themes, but given that we do not visibility into these through the features alone, we chose the ratings as a second-best proxy.

After the k-means algorithm was run and the cluster centroids calculated, we pick the sentence closest to the centroid as the exemplar summary. We therefore always generate five sentences as extractive summaries of the reviews for each product, one for each cluster.

4.4.2 Affinity Propagation Clustering

Affinity Propagation does not require the number of clusters to be determined ahead of time, which makes it very suitable for our task of clustering review sentences.

The algorithm iterates by updating two matrices a (availability) and r (responsibility)

First, the responsibilities matrix, which reflects how good of a point *k* is to be an exemplar for point *i*, is updated according to:

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\}$$

The function $s(i,k')$ is the negative euclidian distance squared between the two points.

Then the availability matrix, which reflects how appropriate it would be for point *i* to choose *k* as its exemplar, is updated according to:

$$a(i,k) \leftarrow min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} max(0, r(i',k))\right) for\ i \neq k\ and$$

$$a(k,k) \leftarrow \sum_{i' \neq k} max(0, r(i',k))$$

The iterations are performed until the boundaries remain unchanged for 15 iterations.

In our case, each k represents a sentence in our reviews, and the i^{th} sentence is the cluster center. Distance is calculated using the negative squared distance of 2 data points in 128-dimensional space (nnlm).

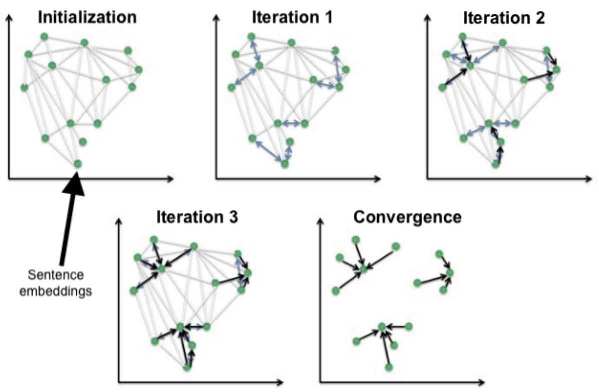


Figure 3: Affinity Propagation [10]

4.4.3 DBSCAN Clustering

Unlike k-means and Affinity Propagation, DBSCAN [11] is a density-based clustering algorithm that does not base its clusters on a single centroid point or value, but rather groups together points that share regions or neighborhoods of density such that all the points in this neighborhood are reachable via a chain of nearby points that are each less than a certain maximum distance away from each other. This maximum distance is a configurable parameter referred to as epsilon (ϵ). If a data point has no points that are less than a distance of e away from it, it is considered an “outlier” and thus can be thought of as belonging to no cluster or belonging to a cluster of size one. While DBSCAN implementations often use Euclidean distance as the default means of determining the distance between two vectors, we found cosine similarity a more effective at producing a larger number of distinct clusters. For our value of ϵ, we used 0.2. Although DBSCAN doesnt have a concept of an exact center of a cluster as k-means and affinity do, there is a still a notion of core points. Whether a point is said to be a core point is determined by the number of neighboring points that fall within ϵ distance away. If this number is greater than *minPoints*, then it is a core point. This value is configured to be any value 2 or greater. In our implementation, we set *minPoints* to be 3. Given that DBSCAN produces at least one core point per cluster, yet our application requires the clustering algorithm to return one representative data point per cluster, core points with larger values of n were preferred.

4.4.4 PageRank

PageRank was developed by Google as a method to rank search results. The algorithm has also been used for extractive text summarization as first proposed by Mihalcea et al. [12] under the name TextRank. Our model here is conceptually inspired by TextRank. In TextRank, we model fundamental units of text as vertices in a graph which are connected to one another with an edge weight. The edge weight quantifies the similarity of the pair of nodes. The nodes can be any unit of text we want to analyze. We choose sentences from reviews. For each product, we split every review into its constituent sentences using sentence tokenization. Then every sentence becomes an individual node in the PageRank graph. We calculate the edge weights of the graph via the cosine similarity of the sentence embeddings for every sentence pair. We, therefore, construct an adjacency matrix for the graph (no self-connections). Now PageRank can be used to rank the vertices of the graph i.e. the sentences. Our main contribution versus the original TextRank algorithm is the use of sentence encoders and using cosine similarity (adjusted so that values are between 0 and 1) between densely represented sentence vectors as the elements of the graph adjacency matrix. When the sentences are ranked by PageRank, we can just choose the top sentences until we reach the maximum number of words/sentences allowed in our summary (human input). Finally, TextRank has a tendency to significantly favor ranking very long sentences the highest. However, this is not what we want in a summary. The authors of the original paper handled this by normalizing scores by the length of sentences. We use a simpler greedy algorithm approach. We loop through our ranked sentences in descending order and measure the length of every sentence against a hard maximum word count. If the number of words in the candidate sentence summary is less than or equal to this maximum, we include the sentence in our extractive summary otherwise we move on to the next highest ranked sentence. We do this until we reach the number of sentences which we specify our model must produce.

4.5 Unsupervised Abstractive Summarization

The above section covered extractive summarization methods where we simply extracted sentences from all reviews which our model determined would represent all product reviews in the most complete and non-redundant way possible. However, it would be really interesting to be able to not only extract summarizing sentences but at the same time accurately paraphrase the reviews. For example, if there are two extractive sentences which are selected, then it is possible that the sentences could be combined in some way to generate even more compact, succinct and globally coherent summary. This is also a common way that humans might perform summarization. This task is called abstractive summarization where we have the ability to summarize a document (or multiple documents) using words that were not necessarily present in the document but have the same meaning and might fit better in the global context. Given this problem entails natural language generation, it is inherently challenging with the added difficulty that we are trying to attempt this in an unsupervised manner. We finished implementing an abstractive summarization model for summarizing product reviews. Our approach closely follows that of Chu et al. [4]. A basic overview of the model is as follows. It is split into 2 halves: The first half, called sequence to sequence autoencoder, contains an encoder (ϕ_E) and a decoder (ϕ_D). The encoder takes in k product reviews and generates a review representation for each of the k reviews. The decoder then aims to reconstruct each review from the encoded representation. The training objective is to minimize the average reconstruction error which is quantified via the cross-entropy loss.

$$\ell_{rec}(\{x_1, ..., x_k\}, \phi_E, \phi_D) = \sum_{j=1}^{k} \ell_{cross\,entropy}(x_j, \phi_D(\phi_E(x_j)))$$

The second half, called summarization module, averages the k encoder generated review vector representations from the first half and tries to generate a single sequence using the decoder parameterized with the same weights from the above autoencoding step. This generated sequence is meant to be the summary. This candidate summary is then encoded again using the same encoder module from the sequence autoencoder to give a summary embedding. We then calculate the cosine distance from this summary embedding to each of the k product review embeddings which were output from the encoder of the autoencoder.

$$\ell_{sim}(\{x_1, ..., x_k\}, \phi_E, \phi_D) = \frac{1}{k} \sum_{j=1}^{k} d_{cos}(z_j, \phi_E(\phi_D(\overline{z})))$$

$$\ell_{model} = \ell_{rec} + \ell_{sim}$$

We use $k = 8$ in our experiments. The basic intuition is that the autoencoder loss serves to constrain the generated sequence in the language domain while the cosine loss tries to enforce the content preservation aspect of the summary. As is common in RNN/LSTM training, we train the sequence autoencoder using teacher forcing while at test time we use a greedy decoding strategy where at each time step we pick the highest probability word and use that as input to the LSTM to generate the next word and so on. We truncate our sequence to have a maximum length of 100. For the decoding section of the summarization module, we cannot use teacher forcing while training as we do not have ground truth summaries and therefore use the Gumbel-Softmax trick [13] [14], which approximates sampling from a categorical distribution (in this case a softmax over the vocabulary) but at the same time allows gradients to be backpropagated through this discrete generation process. This process is expected to converge to a set of weights which would give us a coherent and comprehensive abstractive summary. We tested out this abstractive model in isolation on raw product reviews.

The abstractive summarization model was trained on 10,000 products. We selected products which had at least 50 reviews and also removed those which had more than 114 (90th percentile) reviews. We randomly chose $k = 8$ reviews from every product and had about 80k reviews to train on and tested on 1.5k reviews. We trained for 5 epochs, used 2 GRUs (one for the encoder and another for the decoder) each with hidden size 512. We truncated our vocabulary to the 20k most common words in our corpus and decided to train embeddings from scratch. We used 300-dimensional word embeddings. Our input reviews were also truncated in length to a maximum length of $L = 100$. For the Gumbel-Softmax we used an initial temperature of 2 and annealed the temperature with a decay rate of 0.96 every 100 steps. Finally, we used a learning rate of 0.001, Adam optimizer, and our model was implemented in Tensorflow.

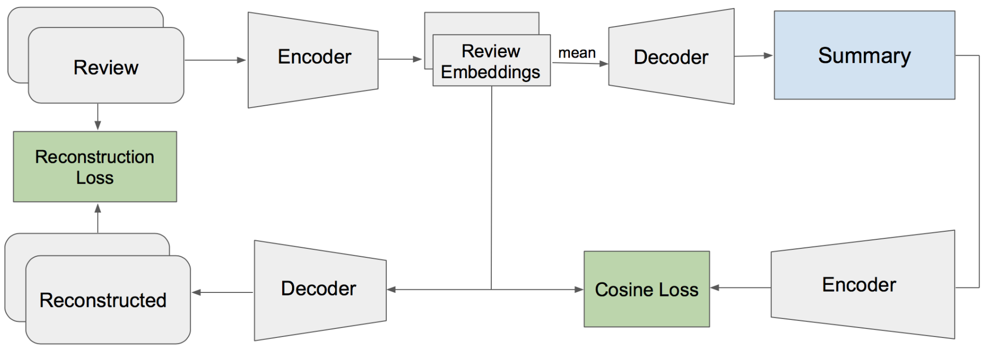


Figure 4: Unsupervised Abstractive Summarization Architecture

5 Results and Discussion

Model	ROUGE-1 (/100)	Semantic Similarity (-1 to 1)	Sentiment Accuracy (/100)	Attribute Match	Content Preservation
PageRank	21.7	0.96	64	3.72	3.93
PageRank†	21.4	0.99	63	3.62	3.80
Affinity	22.1	0.94	64	3.58	3.67
Affinity†	21.3	0.98	60	3.64	3.56
DBSCAN	16.7	0.79	63	3.60	3.85
K-means†	20.5	0.99	61	3.56	3.42
Baselines					
K-means	21.3	0.94	66	3.48	3.44
Random	18.5	0.97	57	3.02	2.33
Abstractive	30.2	0.76	52	3.36	3.22

† Uses *word2vec* embeddings, otherwise *NNLM*

Table 1: Comparison of extractive summarization results

5.1 Efficacy of Metrics

Our human evaluation metrics provided a useful foil to our automatic metrics. Since the ultimate goal of summarization is to provide summaries that are useful to humans, our manual evaluation metrics provided a way for us to assess the meaningfulness of our various automated metrics. For

Sentiment Classifier	Train Accuracy	Val Accuracy
Amazon Reviews w/ 3 Classes	91.4%	81.2%
IMDB Reviews w/ 2 Classes	98.4%	87.7%

Table 2: Performance of Sentiment Analysis

our machine evaluation metrics, our preferred metric is *sentiment accuracy*. As seen in Table 1, it correlates well with our human evaluation metrics. ROUGE-1 is also a fair evaluation metric, which reflects ROUGE-1 favoring longer sentences with more variety, which as a heuristic can mean a better summary. However, as we continue to improve the summary, the usefulness of ROUGE-1 starts to decrease because better summaries would all have good coverage of words and variety and salience starts to matter much more. Sentiment analysis gave us the best correlation to human results. It gives better coverage of salience since the sentiment of the generated reviews should match the original reviews.

5.2 Extractive Summarization

The results shown in Table 1 were computed by running the algorithms on 1000 products in the electronics category of Amazon reviews dataset which had anywhere between 50 and 100 reviews each. Affinity Propagation and PageRank algorithms were the best performers when used with the NNLM sentence encoder, according to our human evaluation metrics, which closely reflects the proxy for *sentiment accuracy*.

PageRank with NNLM performed the best with a ROUGE-1 score of 21.7, *sentiment accuracy* of 64, attribute match of 3.72, and *content preservation* of 3.93. Affinity Propagation is the second best performing model with ROUGE-1 score of 21.4, sentiment accuracy of 64, attribute match of 3.58 and content Preservation of 3.67.

Since PageRank essentially ranks each sentence based on its importance (as measured by similarity) with all other sentences, the resulting sentences picked were higher quality, albeit a bit similar as seen in Table 4. For example, it missed the fact that the camera consumes battery very quickly, which was captured by the k-means model. The other downside to the PageRank Model is it does not cluster, so we do not get a report of how many users wrote something similar, although this can be simulated by using reporting how many of the other unique reviews produce a cosine similarity past a threshold.

K-means and Affinity Propagation both cluster and we are able to produce a count of how many users wrote a similar sentence, so we see more variety such as discussing zoom quality and battery life for the Canon A300 Camera. Though currently the algorithm simply picks the sentence which is closest to the centroid as the most exemplar sentence, which can lead to sentences that do not make sense when taken out of context.

Across the board NNLM encoder performed better than Word2Vec, this suggests that the quality of the encoding has a significant impact on the summarization model - indeed, since we are operating on the sentence encodings, a poor encoding would surely lead to a bad result. In this case, the NNLM encoder actually takes the order of the words that precede it into account to predict the next word, whereas Word2Vec uses a bag-of-words model that only takes into account the existence of words, but not the ordering. More sophisticated encoders such as Universal Sentence Encoders [15] and ELMo [16] are expected to do even better, especially if they are fine-tuned to the dataset in question using unsupervised language model training.

We note that our baseline, using k-means with NNLM has a higher Sentiment Accuracy than both PageRank and Affinity Propagation. This is likely due to our use of $k = 5$ for k-means, which does a better job of choosing a centroid that is an average of all reviews.

5.3 Abstractive Summarization

Abstractive summarization (ABS) actually gave the highest ROUGE-1 score compared to all other extractive methods. However, one must note that the ground truth for ABS was actually only the $k = 8$ reviews randomly chosen as input into the model and not all product reviews. We justify this since, unlike the extractive algorithm, the abstractive model does not see every review and therefore it would be unfair to judge its performance on being able to summarize information it never read. Besides this, our other forms of quantitative evaluation like sentiment consistency and semantic similarity were 52 and 0.76, respectively. Also on our two qualitative metrics, ABS was better than random selection.

One of the chief observations we had during qualitative evaluation, was that while the model was reasonably good at generating human understandable text, the content preservation, and intra-review sentiment consistency was relatively poor. This could be due to a mixture of a few main causes:

1. We noticed some interesting examples, where the content preservation problem could have been mostly caused by our limited fixed vocabulary. By the nature of product reviews, there may be many words, like the name of the product, specific product type etc., which while centrally important to the product summary, are not that common in the overall corpus and so get dropped out in the count-based vocabulary truncation. For example, there was an example of a Samsung smartphone and the reviewer mentioned the camera. However, our summary generated listed Cannon as the company along with the word camera. Then, towards the end of the summary, it also referenced the screen size of the phone which was the topic of one of the other 8 reviews. This issue could potentially be solved by Pointer generator networks [17]. Note that even if these words are present in the corpus but have very low frequencies, the problem might still exist as the model does not see them in context as often.

2. Cosine distance might not be the best metric for content preservation and consistency. We see this in some reviews where the first part of the summary was consistent content while the second was talking about a different product altogether.

3. Additionally, we did not do any form of segmentation before inputting into the summarization model based on themes or sentiment. For example, if our $k = 8$ inputs to the abstractive model contained four reviews with rating 1 or 2 and four with rating 4 or 5, then inherently our summary would not sound consistent. Same issue with themes/reasons that people might give for their reviews and competing products. Therefore, some sort of automated segmentation before this would potentially help a lot, especially when it comes to the sentiment accuracy mentric (which was the lowest relative to the extractive methods). One idea is to use our extractive summaries as input to the abstractive model. However, note that some of the topics we talked above like consistency of summary and reasoning behind ratings are not explicitly forced to be consistent in the extractive methods as well.

6 Error Analysis

For our extractive review summarization, errors can be categorized as relating to an issue with how review sentences are clustered, or errors relating to what we can call *exemplars*: the chosen review sentence from the cluster as being representative of that cluster such as the sentence with the lowest distance to the centroid.

6.1 Clustering Errors

Given that an optimal clustering arrangement consists of a modest number of clusters (5-10 for example) that identify the most commonly occurring review themes, we analyzed shortcomings in each of the clustering algorithms based on when and why they failed to achieve this.

Certain clustering algorithms such as DBSCAN often produced a cluster count that was too small because of a tendency to group data points as either in the main cluster or as outliers, often resulting in 1-2 large clusters with many outliers. Even with careful tuning, DBSCAN often produced a low number of clusters that contained review sentences that upon human inspection could logically be thematically grouped with greater granularity.

For the clustering algorithms for which we were required to set a cluster count in advance such as k-means, errors occurred at times as a result of the body of reviews inherently having less salient themes than the preset cluster count. This could result in clusters that contained reviews that either had a hodgepodge of review sentences that differed in theme or in small clusters with review sentences of low relevance.

6.2 Exemplar Errors

Given a well-organized set of clusters, many errors could still arise from issues with the chosen exemplar. For instance, given a cluster containing many review sentences that relate to a certain product feature, the exemplar may also pertain to this feature, but with a sentiment that does not reflect the overall sentiment applied to this feature in the cluster. Given that the clustering algorithm may be heavily influenced by content words, it is not guaranteed that the sentiment commonly used when referred to this content may also match.

Similar to the above, but less common, is the exemplar not referring to the product feature most commonly described in the cluster. A cause of this may be if there are many review words in the cluster that do not directly relate to a product feature but skew the cluster in a way that leads the exemplar to be a review sentence that refers to a product feature not commonly found in the cluster.

Another shortcoming we encountered with extractive summarization were cases in which the summary sentence referred to multiple things. While clustering at the sentence level means that the exemplars are usually fairly concise, there are still cases where the exemplar sentences refer to multiple things, such that only part of the sentence is representative of the main themes found in the cluster. For example, "So for the burner its 5 stars - maybe the software will work with my old Windows ME box when I get around to trying it out..."

Since our summary sentences are extracted from full multi-sentence reviews, summaries on occasion produced out-of-context errors where the sentence would refer to something previously written in the full review. E.g. "Unlike Kodak, which has provided me with 4 coasters out of 15 used."

6.3 Abstractive summarization

With abstractive summarization, the most common issues we encountered were consistency and content preservation errors. While the summaries produced nicely generalized language that captured the high-level content and sentiment of the reviews, the distinct points referred to in the summaries were not always easy to differentiate. In addition, we encountered cases where a summary gave two contradictory statements about a product, such as "the sound quality is great but the sound is not good" which was rare to non-existent in our extractive summaries. To a lesser extent another issue we encountered was repeated phrases, such as "the controls are clear and crisp and crisp and crisp."

7 Conclusion

We proposed both extractive and abstractive systems for summarizing Amazon reviews using unsupervised learning. We also established a set of manual and automated metrics for evaluating our models in the absence of human annotated ground truth.

We found the best performing model for extractive summarization is PageRank, with a content preservation score of 3.93, though the model is unable to report the number of people who wrote something similar. The best model which does include statistics on how many people wrote the same was Affinity Propagation. Abstractive summarization was able to yield human readable sentences that talk about the product, though content preservation was a challenge as sometimes the model appears to lose its train of thought and starts talking about a totally different product.

Much work is required to improve multi-document summarization to a point where it is acceptable for use by the general public, especially in the purely unsupervised setting. First, our human evaluation metrics are still the gold standard for measuring summarization performance and there needs to be a scalable way to do such evaluation, perhaps through Mechanical Turk, but more preferably through machine evaluations. Second, better machine evaluation methods could be a good proxy for these manual metrics; there should be coverage for both breath and salience. For the extractive case picking the correct exemplar sentence when given a cluster of sentences can be much improved, such as using rules that avoid picking sentences that start with conjunctive adverbs (E.g. "however", "therefore", etc.), so that sentences are representative and do not seem out of context. And finally, for the abstractive case, coming up with better metrics for maximizing content preservation than the simple cosine similarity metric used today remains an open area of research.

References

[1] G. Rossiello, P. Basile, and G. Semeraro, "Centroid-based text summarization through compositionality of word embeddings," *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*, 2017.

[2] D. R. Radev, H. Jing, and M. Budzikowska, "Centroid-based summarization of multiple documents," *NAACL-ANLP 2000 Workshop on Automatic summarization* -, 2000.

[3] K. Yatani, M. Novati, A. Trusty, and K. N. Truong, "Analysis of adjective-noun word pair extraction methods for online review summarization," *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011.

[4] E. Chu and P. J. Liu, "Unsupervised neural multi-document abstractive summarization," *CoRR*, vol. abs/1810.05739, 2018.

[5] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Analysis of adjective-noun word pair extraction methods for online review summarization," *ICLR*, 2018.

[6] J. Mcauley, C. Targett, Q. Shi, and A. V. D. Hengel, "Image-based recommendations on styles and substitutes," *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 15*, 2015.

[7] "Token based text embedding trained on english google news 200b corpus," *TensorFlow Hub*, 2018.

[8] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3:1137-1155, 2003.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," pp. 3111–3119, 2013.

[10] W.-C. Hung, C.-Y. Chu, Y.-L. Wu, and C.-Y. Tang, "Map/reduce affinity propagation clustering algorithm," *International Journal of Electronics and Electrical Engineering*, vol. 3, no. 4, 2014.

[11] A. Ram, S. Jalal, A. S. Jalal, and M. Kumar, "A density based algorithm for discovering density varied clusters in large spatial databases," *International Journal of Computer Applications*, vol. 3, p. 14, Oct 2010.

[12] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

[13] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017.

[14] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *CoRR*, vol. abs/1611.00712, 2016.

[15] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," *CoRR*, vol. abs/1803.11175, 2018.

[16] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *CoRR*, vol. abs/1802.05365, 2018.

[17] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," pp. 1073–1083, 2017.

A Appendix

Input: Amazon Reviews (91)	Output: 5 Single Sentence Summaries
I did a lot of research before choosing this speaker. Despite audiophiles not recommending it, I chose it because of its compact size, the enthusiastic videos online... $\langle doc \rangle$ After much searching and researching online while trying to decide what type and brand of speakers to use for home theater use... $\langle doc \rangle$ This definitely puts the vocal up front... $\langle doc \rangle$...	I replaced my old center channel speaker with this one and am amazed at the quality of the sound. It has a good sound that I think the average person will be more than pleased with. The JBL's were more $ than the rest and sound that way. I replaced my Pioneer center channel with the Cerwin Vega and couldnt be happier. This speaker sounds great and it doesn't disappear behind the rest of my home theater.

Table 3: Example Input and Output

Model	Review
k-means	[57] "Have owned this camera for a few years.", [51] "The camera has a good zoom on it, and is very easy to use.", [56] "I do not enjoy carrying the larger movie cameras and this is just right for me.", [44] "Personally I love it, but if you want to be able to zoom in on your subjects, definitely pony up the cash and move up to a more expensive camera.", [55] "I can get at least 60 shots using these batteries with the camera in full function."
Affinity	[24]"I wasn't able to take a picture until the next day." [18]"Have owned this camera for a few years.", [14]"I got this camera a couple of months ago and I'm not real please with it.", [15]"The camera has a good zoom on it, and is very easy to use.", [14]"If you need small cameras, you have to typically settle for picture quality that LOOKS like it came from a tourist gadget. Not this one!"
DBSCAN	[72]"I even lowered the master volume on my Pioneer AV system since the sound coming out of this speaker is so much better than my speaker in a box.", [20]"It has good sound and quality for it's size.", [6]"PRISTINE SOUND NEED I SAW MORE OH YEAH THE BASS MADE ME GO BUY THE REST OF THE SURROUND SOUND CERWIN VEGA SYSTEM HMMMMM MIGHT GET THE SUB WOOFER ALSO WE WILL SEE LOL", [3]"Although if you know the sound you're looking for, you may want to shell out a few more bucks."
PageRank	"If you need small cameras, you have to typically settle for picture quality that LOOKS like it came from a tourist gadget.Not this one!", "This is my first digital camera and while I was camera searching there are two very important things that you have to take into consideration.", "I do not enjoy carrying the larger movie cameras and this is just right for me.", "I love it...It is my first digital camera but for the price it really has more features than you would expect.", "Try experimenting with the features to find something that suits you.For the money, its a great camera, but dont expect miracles out of it."
Abstractive	"I have been using the product for a week ago and it works great with my iphone 3g. The sound quality is not too bad. This is a good product. I would recommend it to anyone who wants to keep the phone in the car. Is a good product for a small room."

Table 4: Example summaries for Canon PowerShot A300 camera

B Codalab Worksheet

https://worksheets.codalab.org/worksheets/0x075831de7a0244ee80bf689363f27c88/