Improving the Informativeness of Abstractive Opinion Summarization

Eric van Schaik

Abstract—Although current state-of-the-art abstractive text summarization has improved significantly since the introduction of the transformer architecture, opinion summarization has lagged somewhat behind, partly due to the lack of labeled training data. Various unsupervised learning methods have been tried to solve this problem. Most seem to produce less informative text than regular text summarization, reflecting the lower quality of the opinion summarization datasets. This effect is exaggerated by an apparent lack of metrics measuring this perceived informativeness, resulting in little attention being payed to this aspect of opinion summarization in related research. Therefore, this research will focus on the following research question: How can the informativeness of opinion summarization be improved? First, some ways of measuring informativeness are proposed. Then, a new opinion summarization model is proposed, in an attempt to improve upon these metrics. This new proposed model roughly consists of three parts: *topic modeling*, where a number of relevant review topics are generated, *ranking*, where all the sentences from the input reviews are ordered based on importance, and *filtering*, where only the most important sentences are selected as input for the summarization model. Finally, the model is tested and compared with current opinion and text summarization models, namely the COOP [1] and PEGASUS [2] model, respectively. It is shown that the performance of our model is often closer to the state-of-the-art text summarization than to opinion summarization models, while retaining an accurate sentiment value. Small-scale human evaluation is included as well, its results partly supporting the conclusions drawn from the automatic evaluation. Therefore, the model proposed here can provide a good alternative to existing models, albeit dependent on the context.

Index Terms—Opinion Summarization, Topic Modeling, Natural Language Processing, Transformers

1 Introduction

1.1 Brief Overview

Since the introduction of the transformer architecture in 2017 [3], language models have become significantly better at solving various Natural Language Processing (NLP) problems, such as machine translation or question answering. Improvements have been made in text summarization as well, the goal of which is to produce shorter versions of the input texts while retaining the most relevant information. In particular, more focus has been given to abstractive summarization as opposed to extractive summarization. With extractive summarization, the most relevant words and phrases are simply extracted from an input text and combined to form a summary, while abstractive summarization focuses on generating new original sentences capturing the most relevant semantic information of the input text. Therefore abstractive summarization is seen as a more complex problem. With the introduction of more complex language models, such as the transformer BERT (Bidirectional Encoder Representations from Transformers [4]), this type of summarization has become significantly more effective.

Despite this progress in language models, opinion summarization models have lagged somewhat behind. These models aim to summarize texts that mainly convey opinions, such as texts from social media or from customer reviews. The lack of progress in this field seems to be partly due to the lack of labeled training data. With regular text summarization, large text corpora are available with gold-standard summaries, that summarization models can use to optimize their loss function. Those datasets can be news articles, where journalists often already provide a short summary of their article at the beginning (e.g. the XSum dataset [5]), or scientific papers, where the abstract can be used as a gold-standard summary (e.g. the PubMed dataset [6]).

With opinion summarization, the retrieval of such labeled datasets becomes more difficult. The most common datasets in this domain consist of customer reviews. For instance, for products on Amazon [7] or businesses from Yelp [8]. In these scenarios, when users comment on a product, they typically do not include summaries of their opinion. Sometimes gold-standard summaries are produced, purely for the purpose of training and evaluating opinion summarization models, but those datasets remain small [9].

Because of this difficulty in opinion summarization, most new state-of-the-art models use some form of unsupervised learning to train their neural networks [9]. This makes it harder for an opinion summarization model to recognize which parts of the input text are important for summarization. What makes it worse is that the quality of the input texts varies greatly as compared to the text that is generally used for text summarization (in terms of grammatical and syntactical errors, language formality, etc), so that it becomes harder for language models to capture the semantic information [10]. All this has contributed to a lack of progress and consequently, interest into the field of opinion summarization, as compared to text summarization.

This paper will address some of the problems of the field in two ways.

First, some new ways of measuring the performance of existing opinion summarization models are proposed. This is necessary since the lack of training data means there is also a lack of information on how well a certain model is

performing. These new metrics should give some indication of the performance of a model without labeled data, thereby enabling unsupervised learning.

Second, a new model will be proposed that aims at improving primarily upon those new metrics. Because these new metrics are still tentative, fine-tuning a new model on these new metrics would not ensure that the model addresses the given problems, so unsupervised learning is not yet applied here.

1.2 Problem Description

In order to make the problem more specific, it becomes instructive to look at the output of each kind of summarization model. The state-of-the-art text summarization model PEGASUS from Zhang et al. [2] summarizes an introduction section of the Eiffel Tower Wikipedia page as follows:

> "The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building, and the tallest structure in Paris .<n>Its base is square, measuring 125 metres (410 ft) on each side .<n>During its construction, the Eiffel Tower surpassed the Washington Monument to become the tallest man-made structure in the world ."

This summary generated by a PEGASUS model trained on the CNN/DailyMail dataset [11], and is taken from the HuggingFace Hub [12].

To compare it to summaries produced by current state-of-the-art opinion summarization models, take the following summary, produced by the MeanSum model [9]:

> "Was in the area for a few days and I've heard it's been here. The last time we went there, it was just as bad. Food was very good but not very good. I got the same thing as my first time trying this place. Their fries are good but not great either."

MeanSum generated this summary based on the Yelp dataset [8]. Note the contradictory sentence "Food was good but not very good", which is a good example of how much the language understanding of opinion summarization models have lagged behind regular text summarization models.

MeanSum is a typical case of an unsupervised learning model, where the model tries to generate the "average" review, and therefore these summaries merely reflect the average quality of the opinions written on the internet, or on Yelp in this case. To measure their performance, Chu et al. primarily use the ROUGE score. They collected gold-standard summaries by using Amazon Mechanical Turk, a crowdsourcing platform mainly used to label data, where workers were asked to write a summary that "best summarizes the content and the sentiment of the reviews", but also to "write your summary as if it were a review itself" [9]. The underlying assumption that summaries should be written in the language of an average reviewer has led the summaries to suffer from the same drawbacks as the text of the original reviews, such as the aforementioned variety in text quality. In particular, when comparing the two summaries given above, the summary from MeanSum would seem less informative to a reader, given its amount of personal information and use of relatively simple, generic words (generic words are generally unable to carry as much information as more specific words, although it could very well be the case that the simple words are describing things that the reader is more curious about than the more complicated words). As will be shown in Section 3.2, MeanSum can be seen as representative for the current abstractive summarization field, and it has set a standard for looking for the best "average" opinion of the dataset as the main goal in the field. This paper will argue for a more consumer-centric approach, where the goal is to best inform the consumer of a summary of the general opinion of a product or service.

The problem of current opinion summarization models being uninformative to readers was recognized by Iso et al. [1] as well. They used entropy as a metric to show this, where the amount of *information* contained in the encoding of some text was linked to the *informativeness* to a reader. On average this method proved a step in the right direction. However, simply equating information of a word vector to informativeness to a reader could have its shortcomings as well. Take the following summary generated by the COOP framework for example.

> "Took this bag on a recent trip to Europe and was very happy with it. It is light weight and holds a lot of things. The pockets are great for outside pockets to hold a camera or a spare battery."

When comparing to the MeanSum summary given above, COOP summaries seem to be an improvement in that they convey more information. A shortcoming would be that some of the information given at the beginning of the summary is not as informative for a consumer (a consumer likely would not care about the recent trip to Europe). Therefore, a more nuanced look at what would make a summary informative to a reader, needs to be established.

In short, this paper will attempt to address the problem of current opinion summarization models producing sub-optimal summaries when looking at informativeness to the reader. Put another way, this paper will attempt to bridge the gap between the textual quality of state-of-the-art text summarization models and state-of-the-art opinion summarization models. Here it must be noted that this search for more informative summaries should not be at the expense of the retainment of sentiment value. The conveyance of the correct sentiment value is what differentiates opinion summarization from text summarization, so this must be taken into account when focusing on generating more informative summaries. Consequently, this will be taken into account when addressing the previously stated problem.

1.3 Research Questions

The problem described above is addressed by the following research question:

Main Research Question:
How can the informativeness of opinion summarization be improved?

This question can be divided into the following subquestions:

Research Question 1: How can the informativeness of text be measured?
Research Question 2: How can the informativeness of existing opinion summarization methods be improved?
Research Question 3: How can the sentiment value be maintained when focusing on informativeness?

The first two subquestions reflect the two important steps that need to be taken to answer the main research question, while the third reflects an important condition under which the other research questions need to be answered.

To answer the first subquestion, this paper will propose a new set of metrics by which different aspects of informativeness are measured. As the term "informativeness" is not well-defined, and since this research falls within the field of computer science, not psychology, the validity of these new metrics as correctly measuring "informativeness" to a reader will not and cannot be thoroughly checked, and it will take some assumptions to accept the use of these new metrics. This paper will attempt to substantiate these assumptions as best it can, however large-scale human evaluation would be necessary to fully validate the results of this research, and such experiments fall outside its scope. Therefore, the results of this approach should be taken as a step in a new direction for the field of opinion summarization, not as hard evidence that this is the "better" approach.

The answer to the second subquestion will involve the proposal of a new opinion summarization model, using elements of existing text and opinion summarization methods and adding some original elements. Some design choices are substantiated by results of intermediate experiments, where the design choices are given in the methodology section, while the results of these experiments are given later in the results section.

By answering these research questions, this paper presents a new, consumer-centric approach to the problem of opinion summarization, as an alternative to the more mathematically-driven approaches of most papers in the field.

The rest of this paper is structured as follows. First, a scientific background is presented to better understand the problem, as well as some concepts behind the proposed solution. Second, a related work section is presented, going more specifically into relevant papers in related fields. In Section 4, the methodology of this project is laid out, which can be seen as the proposed solution to the given problem. Section 5 discusses the technical details of the experimental setup, used to answer the research questions, while Section 6 presents the actual results of the research experiments. In Section 7, the results are further discussed, and the paper concludes with Section 8, in which the contributions of this research are summarized.

2 Scientific Background

2.1 Problem Background

2.1.1 Recurrent Neural Networks

Before the introduction of the transformer architecture, the Recurrent Neural Network (RNN) [13] and its derivatives, such as the Long-Short Term Memory networks (LSTMs) [14], were commonly used for text summarization and Natural Language Processing (NLP) tasks in general. The RNN is a type of artificial neural network, and as such is designed to generate a number of output values, based on an independent number of input values. In its simplest form, an artificial neural network accomplishes this by multiplying the input values by a certain number of weight matrices. This can be visualized by a graph, such as shown in Figure 1.

Each weight matrix represents all the edges between two adjacent layers in the graph. In the figure, the input values propagate from left to right through the network, being multiplied by the weights between the layers, and resulting in a number of output values. The layers in between the input layer and output layers are generally called "hidden layers" (only one is shown in the figure for simplicity). The term "neural network" comes from a resemblance to a common conceptualization of a human brain, where the nodes are likened to neurons and the edges are seen as connections [16].

Finding the appropriate weights for such a network is typically reformulated as minimizing the loss function of the network, given a number of training examples. These examples consist of valid input/label pairs, and the loss function measures the difference between the ground-truth label and the predicted label. Many loss functions are used in practice, however for illustrative purposes consider the Mean Squared Error (MSE) [17] given below.

$$MSE(Y, \hat{Y}) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2, \tag{1}$$

where Y is the ground-truth label, $\hat{Y}$ is the predicted label, and n is the number of dimensions of the labels.

Training the network would then consist of iterating through the following steps. First, the network is initialized with random weights. Second, a training example is pushed through the network by multiplying the training example with a weight matrix at each layer. Third, the MSE is calculated, and fourth, the derivative of the loss function with respect to each weight is calculated. Fifth, that derivative is used to nudge the weight into the right direction, since the derivative represents the influence a weight has on the loss function. By proceeding through these steps for each training example, the loss function is minimized.

The key contribution of the RNN, as first proposed by David Rumelhart in 1986 [13], is that the value of the hidden layers propagates not only to the output layer, but also to the hidden layers of next iterations. While with the simple neural network, only one input vector was considered, the RNN takes in a sequence of input vectors, and assumes that the value of a previous input vector has some bearing on desired interpretation of the next input vector. See Figure 2 for an illustration of an RNN in its simplest form.

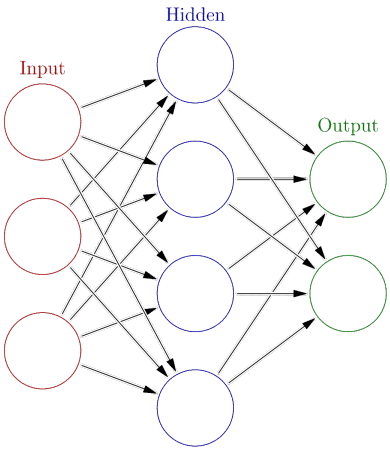

Fig. 1: Neural Network [15]

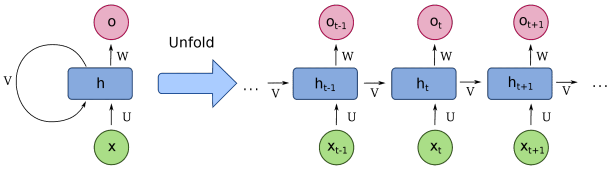

Fig. 2: Recurrent Neural Network [18]

Here the x_t is the input vector at time t (consisting of only one value for simplicity), o_t is the output vector at time t and h_t is the hidden state at time t. Here it is shown that for the calculation of the hidden state at time t, both the value of the current input layer is taken into account, as well as the previous hidden state.

Because of the nature of natural language, the RNN has been the model of choice for many NLP tasks before the advent of the transformer architecture. With natural language, the meaning of one word often greatly depends on the preceding words, so the RNN was a natural fit, since the value of one iteration influences the next. However, with RNNs there is one major drawback: the vanishing gradient problem [19]. This problem is not exclusive to RNNs, as it generally occurs when a neural network gets sufficiently complex. Taking the simple neural network of Figure 1 as a reference, adding more hidden layers would make the influence of weights in the first layers on the output layer much smaller. Therefore, during backpropagation, when the weights are updated according to their influence on the loss function, the weights in the first layers change relatively little, and training these layers of the network becomes unfeasible. With RNNs, this becomes an even larger problem, as the weights of previous iterations need to be updated as well. This problem can be seen as a trade-off, where making the hidden layers more complex would result in the influence of previous hidden layers decreasing. Since the meaning of natural language is complex and is determined by a significant number of previous words, this has been the largest drawback of RNNs in NLP.

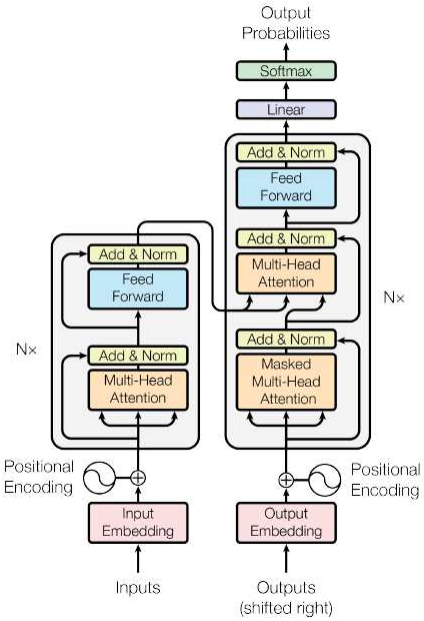

Fig. 3: Transformer Architecture [3]

2.1.2 *Transformers*

The transformer is a new deep learning method, developed in 2017 by Google [3], and has become one of the most popular neural network architectures for many NLP tasks. It adopts an encoder-decoder architecture, and its most prominent feature is its use of attention mechanisms. Where some previous state-of-the-art NLP architectures also made use of attention mechanisms, most notably the LSTM, they mostly relied on a form of recurrence. As the name of the paper by Vaswani [3] ("Attention Is All You Need") suggests, the transformer architecture lets go of this dependence on recurrence and relies solely on attention mechanisms.

The transformer architecture is shown in Figure 3.

The architecture can be divided into an encoder on the left side and a decoder on the right side. Instead of dealing with a sequence of input tokens one token at a time and propagating intermediate values to next iteration,

the transformer deals with the whole sequence of inputs at once. In the figure, this sequence of inputs goes into the transformer in the lower left corner. Each input is first converted into an input embedding, which can be thought of as a form of compression, making the rest of the processes more efficient. Then positional encoding is added, so that the encoder can distinguish between a word that is at the start of the sentence and one that is at the end. This sequence of embeddings is fed into the encoder, which performs multi-head attention on them. Each attention head performs the attention function, which is given below.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

Here the matrices Q, K and V are all obtained by multiplying the input embeddings with a different learned weight matrix. Conceptually, the K matrix is thought of as the Keys, the Q matrix is thought of as the Queries, and the V matrix is thought of as the Values, whereby the attention weight a_{ij} is given by the dot product between the query vector q_i and the key vector k_j, and represents the *attention* that token i pays to token j. The Value matrix then determines how valuable that connection is.

The attention mechanism is performed multiple times in parallel within the same layer, hence the name Multi-Head Attention. With this, the attention layer can focus on different important relationships between parts of a sequence. Intuitively, given a sentence, this could mean one attention head could focus on the relationship between a noun and its adjectives, while another attention head could focus on the relationship between a subject and its verb. In reality, these attention mechanisms quickly become much more complex.

To fully understand the transformer architecture, two more nuances to the attention mechanism need to be explained. First, in the figure it is shown that the decoder starts with performing Masked Multi-Head Attention. This is because the decoder takes as input the whole output sequence. Since the output sequence has only partially been predicted at the time of decoding, the part of the output sequence that is not yet predicted needs to be taken out of the equation. This is done by partially masking the output sequence.

Second, the model makes a distinction between self-attention within the encoder and the decoder layers, and encoder-decoder attention, which is performed when the encoder feeds into the decoder. Everything explained above concerns self-attention, and the only difference between it and encoder-decoder attention is that the value matrix V and the key matrix K come from the encoder, while the query matrix Q comes from the previous decoder layers. That way, the decoder can base its focus partly on the already generated output.

By focusing not only on the previous token in a sequence but on the whole sequence at once, the transformer architecture has managed to largely get rid of the vanishing gradient problem [20], but has also made the generation of one output token much more complex. The impact of this complexity on computational time is greatly reduced by its ability to be parallelized. Most of the calculations in the encoder can be done from the start, and many of the calculations in the decoder can be done when not all of the output embeddings are ready. All this has contributed to the success of the transformer architecture in NLP tasks.

BERT: The Bidirectional Encoder Representations from Transformers, or BERT, is one of the most prominent pre-trained language models using the transformer architecture. Created and published in 2018 by Google [4], it is trained on approximately 3 billion words, and the base version of the model consists of 12 transformer blocks and 110 million parameters. It is trained on only two tasks, namely next sentence prediction and language modeling (i.e. predicting masked words in context), which is why the model only needs an encoder, as the previously predicted tokens are irrelevant with these two tasks. By training only on these two tasks, BERT has become a successful general language model. By using BERT as an encoder and adding some decoder layers, a new model can easily be built and fine-tuned on more specific tasks, such as question-answering or machine translation. This has made BERT one of the most widely used language models in various NLP applications.

2.1.3 Text Summarization

Text summarization is the subfield of NLP which focuses on generating a textual summary given a number of source documents. As mentioned, there are generally two types of text summarization, namely *abstractive summarization*, i.e. generating new original content, summarizing the most relevant information from the source documents, and *extractive summarization*, i.e. extracting the most important content from the source documents. The textual units that are extracted are arbitrary, although sentences are commonly used [21]. To illustrate the difference between these two types of summarization, take the following example [22]:

> "Peter and Elizabeth took a taxi to attend the night party in the city. While in the party, Elizabeth collapsed and was rushed to the hospital."

An extractive summary of this text could be:

> "Peter and Elizabeth attend party city. Elizabeth rushed hospital."

In this example, the textual units are assumed to be words, since sentences would make the example much larger and likely less clear.

An example of an abstractive summary would be:

> "Elizabeth was hospitalized after attending a party with Peter."

In the provided example, it seems like the extractive summarization method has succeeded in finding the most important words in the source text. However by simply appending them, the output summary becomes less readable and more ambiguous, since now it is not clear whether Elizabeth *rushed* or *was rushed* to the hospital. The abstractive summarization method doesn't have these drawbacks, and still succeeds in capturing the most important semantic

information from the source text. Given that extractive summarization often uses sentences as extracted units, these drawbacks are somewhat exaggerated. However, a significant decrease in coherence from one textual unit to the next can be expected, resulting in a decrease in readability and clarity (the ambiguity would likely be less of a problem).

Before the introduction of the transformer architecture, abstractive summarization was largely unfeasible. To quote a text summarization survey paper from 2017, "there is no completely abstractive summarization system today" [23]. Nowadays, a large number of transformer-based abstractive summarization systems are available. HuggingFace has developed an user-friendly API to download and use these models [12], and on their website https://huggingface.co/, a quick search for text summarization models already offers 562 models to choose from. The models differ in a number of aspects, notably their network size. The more parameters a network is given, the better summaries it can produce but the longer it takes to run. How to best handle this trade-off depends on the context, so some variety in models is offered.

Datasets: An important aspect where various text summarization models differ is the dataset they are trained on. One popular dataset is the CNN/DailyMail dataset [11]. This dataset contains approximately 300.000 news articles from CNN or DailyMail, where each news article is provided with a "highlights" section, a short text naturally written by the journalist to draw the reader in. This section provides the text summarization models with a gold-standard summary, ideal for fine-tuning.

Another dataset with this feature is the XSum dataset [5]. This dataset contains approximately 230.000 news articles from the BBC, with the most notable difference that the human-written summaries are no longer than one sentence. This leads to models trained primarily on the XSum dataset producing shorter summaries than its CNN/DailyMail trained counterparts.

Another notable dataset is the arXiv dataset [6], consisting of 200.000 scientific articles, mainly in the STEM fields. Here the abstract of each article is used as a gold-standard summary, and therefore also provides a large source of naturally formed labeled datasets for text summarization. The PubMed dataset [6] is a similar dataset consisting of medical research papers.

As a final example, the public sector often provides articles with human-written summaries. There is the BIGPATENT dataset [24], consisting of approximately 1.3 million patent documents. Each US patent claim needs an abstract to be taken into account, so this also provides good human-written summaries. Similarly, the BillSum dataset [25] provides around 20.000 US Congressional and California state bills, along with a human-written summary as well.

All these datasets show that for regular text summarization, there is no lack of labeled data, and this has made the advances in abstractive text summarization in the last couple of years possible.

Performance Metric: A popular set of metrics for text summarization evaluation is Recall-Oriented Understudy for Gisting Evaluation, or ROUGE [26]. This metric measures the word overlap between a generated summary and a gold-standard summary. There are a number of different ROUGE metrics, the most commonly used being the ROUGE-L score, measuring the Longest Common Subsequence, and the ROUGE-N score, measuring the amount of overlap of n-grams. For unigrams, this score is called the ROUGE-1 score, for bigrams, the ROUGE-2 score, etc. This metric already exists since 2004, and has been used for both extractive and abstractive summarization. It seems to be ill-equipped to take into account the added value of abstractive summarization over extractive summarization. Again taking the example given above, the increased coherence of the text would not be captured by the word overlap metric. Additionally, the ability of abstractive summarization to generate new text that could capture the meaning of the original text with fewer words could even be punished (depending on the reference summaries). Nevertheless, no widely adopted alternative to this metric seems to exist in the text summarization field.

2.1.4 Opinion Summarization

Opinion summarization is a type of text summarization where there is generally more emphasis on the retainment of a texts its sentiment value. A typical use case is the summarization of customer reviews, where a customer or producer of a product wants to know how people generally feel about the product. Text summarization is usually concerned with retaining the most important facts and information from the input text, while in this case it is especially important that the resulting summary retains the same sentiment value as all the input reviews. It is also often important in opinion summarization that the summary is aspect-specific, e.g. the customer wants to know what other consumers think about a certain feature of the product.

In the rest of this section, the field of opinion summarization is laid out based on a paper by Kim et al. [27]. Kim et al. give a nice overview of the different techniques that are traditionally used in opinion summarization, as well as the different NLP subfields that opinion summarization touches and what the relation of those fields is to opinion summarization. They divide the related subfields of opinion summarization into four areas, namely subjectivity classification, sentiment classification, text summarization and topic modeling.

The goal of subjectivity classification is to differentiate between text that contain sentiment value and text that conveys mainly factual information. This is often used as a preprocessing step for the other opinion summarization steps. By removing the purely factual information from the input text, the accuracy of the opinion summarization can be increased.

Sentiment classification concerns the determination of the sentiment orientation of the input text, and could already be used as a quick statistical summary of a product or aspect (for instance, "80% of the reviews were positive" would be such a summary). Sentiment classification is generally done in two ways, there are the lexicon-based methods and the learning-based methods. Lexicon-based methods generally relies on a sentiment word dictionary, where each word is assigned a certain sentiment value, and the sentiment value of a piece of text is the sum of the sentiment values of the individual words. These lexicon-based methods were the most popular methods in previous

decades, due to its relative simplicity. The simplicity is its main drawback as well, since it can be somewhat limited in the amount of complexity it can capture, and it fails to take the context of each word into account when determining its sentiment value. Nowadays learning-based methods are more often used, which requires more computing power but can fix most of the limitations of rule-based classification/prediction.

The next step is text summarization, which can be divided into extractive summarization and abstractive summarization. The difference has been laid out in Section 2.1.3. This summarization step differs from regular text summarization in that the sentiment orientation of the input text is crucial. The goal of the opinion summarization model is to reflect sentiment polarity of the input text as accurately as possible, and after the subjectivity and sentiment classification steps, the model knows which phrases in the input text are the most central to the whole text with regards to sentiment. By combining this knowledge with regular text summarization techniques, the opinion summaries are generated.

Topic modeling, or feature identification, is an optional part of opinion summarization, and is used to discover a number of abstract topics, given a piece of text. It has a prominent role in the paper by Kim et al. [27], and this shows that it used to be a more central part of opinion summarization (more recent papers rarely include topic modeling in their model, as shown in later sections). The discovered topics were often used to generate a summary for one specific feature of a product. For example, this would enable a retailer to get a summary of what his customers thought of the price of his product, or its durability. Kim et al. go into a number of feature identification techniques, and divide them into shallow NLP-based methods and mining-based methods. Both tend to use rules, although the mining-based methods are more capable of handling more complex input. Shallow NLP methods often use POS-tagging (Part of Speech-tagging, where every word is put into an grammatical category) and parsing to find candidate features, for instance by assuming that all features must be nouns. Mining-based methods tend to be less strict, but often use domain-dependent rules and heuristics for finding features.

These different subfields of opinion summarization do not need to be explicit, as later sections will show. Since the advent of abstractive summarization, and as text summarization models have become more complex, these subfields are often implicitly captured by one model.

Datasets: In the opinion summarization field, two datasets have become the standard for training and testing summarization models. These datasets are the Amazon dataset [7] and the Yelp dataset [8]. The Amazon dataset contains 142.8 million reviews, written between May 1996 and July 2014, and is strictly divided into separate categories, such as "Books", "Video Games", etc. Each review contains, in addition to the review text, a rating, product id, and a customer id, among other things. The Yelp dataset is less structured. It is separated into files containing different JSON objects, with a file containing businesses, a file containing reviews, a file containing users, etc. These files are linked together with IDs, where one review object for instance would contain both a customer id and a business id. Other notable metadata features are the rating in the review file, and the categories in the business file, where Yelp handles the categories aspect more as if it were tags, giving a number of different categories with no clear order, and other businesses having only partial overlap. For instance, one categories instance could be "Gastropubs, Food, Beer Gardens, Restaurants, Bars, American (Traditional), Beer Bar, Nightlife, Breweries", where other businesses could have similar but not identical categories. By structuring reviews in this manner, Yelp offers no simple way of handling reviews on a per-category basis.

For the rest of this paper, opinion summarization will be discussed in the context of summarizing these kinds of customer reviews, as most current opinion summarization research is done with these datasets as well.

2.2 Technologies

2.2.1 Word Vectors

Word vectorization, or word embedding, is the process of translating a word in natural language to a real-valued vector, capturing as much of the meaning of the word as possible. These vectors makes certain calculations with words possible, such as measuring the semantic similarity between two words. This has enabled a number of advances in the field of NLP.

There are two types of word embeddings, namely frequency-based embeddings and prediction-based embeddings. Frequency-based embeddings are word embeddings that are generated by counting how often a word in occurs, and in what context. TF/IDF is traditionally taken as an example of this, although it can be considered an almost trivial example, as this method only produces one value per word, and is therefore able to encode little meaning. In short, TF/IDF is calculated by the Term Frequency (TF), i.e. how often the word occurs in the given document, multiplied by the Inverse Document Frequency (IDF), i.e. the inverse of how often the word occurs in the other documents of the corpus. This gives an indication of how important, or how informative, a word is in a given context.

Another example of frequency-based embeddings which does produce a meaningful vector is the co-occurrence matrix. In its simplest form, this method computes a matrix with all the unique words in the corpus on both axis, and each cell representing the co-occurrence of the words associated with the column and the row, i.e. how often does the word from row i occur next to the word from column j, resulting in the value of cell ij. In practice, these matrices take more than the next word into account, and to remain computationally feasible are also normalized and compressed, however the general idea remains.

More recently, prediction-based embeddings have become the standard for word vectorization. These kinds of methods focus on predicting a word given its context, and its most well-known example is Word2Vec. Word2Vec was proposed by Mikolov et al. [28] at Google in 2013, and it has become a standard in NLP due to its simplicity and versatiliy. Word2Vec consists of a shallow neural network, which is trained with a large corpus of text. This is made possible by the network not retaining the intermediate

results, as opposed to the co-occurrence matrix discussed above. Due to the larger corpus size becoming computationally feasible, the retention in semantic meaning proved to be more effective as well. This was demonstrated by the vector space preserving, in addition to semantic similarity, other semantic patterns as well. For instance, it could be demonstrated that by subtracting the word vector of "man" from that of "brother", and adding the "woman" vector, using regular vector arithmetic, a vector is produced that most closely resembles the "sister" vector in the vector space, thereby preserving those semantic relationships as well. Such features underline the robustness and the versatility of this method.

A research team at Stanford developed an alternative to Word2Vec, called GloVe [29], which does take into account all the co-occurrences of two words. Due to certain matrix factorization techniques, training this model remained computationally feasible, and in practice the performance of both models is similar.

Both models can easily be used through the API of the Gensim library. This library offers a number of pre-trained word vector dictionaries, in addition to methods for training a Word2Vec or GloVe model with one's own corpus.

One notable criticism of these kinds of word embeddings is the fact that they are unable to distinguish between different uses of ambiguous words, since they do not take the context into account after training. The word "pound" will produce one vector with a pre-trained Word2Vec or GloVe model, while the meaning could vary from the British currency unit to the British mass unit. To account for this ambiguity, BERT has increasingly been used to produce contextual word embeddings. However, due to the simplicity of the context-free word embeddings, Word2Vec and GloVe remain popular as well.

2.2.2 Topic Modeling

Topic modeling usually refers to an unsupervised machine learning method, where a collection of documents is scanned for important words and phrases, and is clustered along a number of word groups, representing abstract topics. A popular topic modeling tool is Latent Dirichlet Allocation (LDA), already developed in 2003 by Blei et al. [30], which has become a standard in NLP. Similar to some of the word vectorization techniques, this method builds on the assumption that similar words occur in similar contexts. The context in this case is the whole document that the word occurs in, where each word is taken into account equally. This is called a bag-of-words model, referring to the fact that the order in which the words occur does not matter. In short, LDA works as follows. It assumes a given number of topics, and randomly assigns each word to one of the topics. It then continually updates the probability that a word belongs to a certain topic by calculating both the proportion of words in the document that belong to the given topic (a measure of how likely it is that the document belongs to the topic), and the proportion of documents that belong to the given topic and contain the same word (a measure of how likely it is that the word correctly belongs to the given topic). By continually updating each word with these steps, this model eventually converges towards stable topics. The words with the highest probability of belonging to the topic are chosen to represent the topic.

The main drawback of this method is inherent in all the bag-of-words models, namely that the context of each word is not taken into account when predicting the topic associations. As in many other NLP subfields, attempts are made to tackle this problem also with BERT, as this has proved effective in capturing semantic information from text in general. In particular, BERTopic [31] offers a well-documented and intuitive API for topic modeling with BERT. BERTopic uses a variety of existing methods to produce a topic model, given a number of input documents. It starts by using the `sentence_transformers` package [32] to import a version of BERT called `"all-MiniLM-L6-v2"` [33]. With this model, a vector representation is produced for each document, after which the amount of dimensions of the embeddings is reduced with a method called UMAP [34]. This is necessary for the clustering of the documents, as HDBSCAN [35] (the chosen clustering algorithm) is not able to deal with the high dimensionality of the embeddings produced by BERT within feasible computational limits. To find useful topic representations, TF/IDF is used. Here each cluster is viewed as one document, and by performing TF/IDF on each in the cluster, the most important words are discovered and used as topic representations.

2.2.3 Sentiment Analysis

Sentiment analysis is a subfield of NLP that focuses on predicting the affective value of text. Many use cases can roughly be assigned to one of two forms of sentiment analysis, mirroring the first two steps of opinion summarization laid out in Section 2.1.4, namely the sentiment orientation, or whether a text is positive or negative, and the intensity, or how positive or negative a text is. One popular pre-trained model for sentiment analysis is Valence Aware Dictionary for sEntiment Reasoning (VADER) [36]. This is a lexicon-based method, meaning that each word has an associated set of polarity scores. VADER gives for some text a positive score, a neutral score, a negative score and a compound score. The positive, neutral and negative score can be seen as a probability distribution, where all the scores add up to one, based on the distribution of positive, neutral and negative words in the text. The compound score is calculated as follows:

$$\frac{x}{\sqrt{x^2 + \alpha}}, \tag{3}$$

where x is the sum of polarity scores and α is a chosen constant, defaulted to 15. This score reflects the sum of all the polarity ratings of the individual words.

3 Related Work

In this section, a number of related papers are discussed. Firstly, a popular text summarization model is discussed, which is later used as a baseline model as well. Secondly, multiple relevant opinion summarization models are laid out. Finally, the field of text quality is explored, and various papers attempting to measure it are presented.

3.1 Text Summarization

While BERT is the most prominent transformer-based language model, some derivative models have been designed and trained more specifically for text summarization. PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive SUmmarization Sequence-to-sequence models [2]) is such a model, also developed at Google, and is able to outperform BERT on several text summarization benchmarks. Its most notable contribution is that of a slightly altered pre-training objective. Where BERT uses a Masked-Language model (MLM), PEGASUS chooses a new method called Gap Sentences Generation (GSG), which more closely resembles the final downstream task of summarization. With MLM, BERT performs pre-training by randomly masking certain words from the input text and then tries to predict those masked words (this is called self-supervised training). With GSG, PEGASUS masks whole sentences during pre-training, and chooses these sentences based on importance. It estimates this importance by calculating the ROUGE score between the sentence and the rest of the document, where the masked sentence could be viewed as a pseudo-summary. This pre-training task of predicting a pseudo-summary better resembles the downstream task of text summarization, and therefore PEGASUS performs better in the latter case.

3.2 Opinion Summarization

OpinionDigest is a system developed by Megagon [37], and is a typical opinion summarization architecture, with special attention given to transparency. The core of the model is a transformer that is able to generate a summary, given a number of extracted opinion phrases. These opinion phrases are extracted with Snippext [38], an opinion mining system that is fine-tuned through semi-supervised learning. During the training phase, those opinion phrases are extracted from a given review with a pre-trained tagging model, and the transformer is trained to reproduce the original review with the opinion phrases as faithful as possible. To subsequently generate a summary, all the reviews of an entity are collected, all opinion phrases are extracted and clustered according to their similarity in text, polarity and aspect category, and the most central opinion phrases in those clusters are selected. The opinion phrases are given as input to the trained transformer, which generates a summary. See Figure 4.

This setup is inspired by the MeanSum architecture of Chu et al. [9], while improving on some important drawbacks of the original model. MeanSum has become a standard for opinion summarization models, due to its use of unsupervised learning, enabling opinion summarization without large quantities of labeled data. First they use an encoder to map each review to a vector in a lower-dimensional latent space, in this case by taking the last hidden and output state of an LSTM. Each vector is then decoded by the decoder LSTM to produce a reconstructed review, and the cross-entropy loss between the original review and reconstructed review is used to train the decoder. A summary is produced by taking the mean of all the encoded reviews (so simply the mean of all the final hidden and output state values) and decoding that mean. The encoder is trained by using the summary as input and using the cosine distance between the encoded summary and encoded reviews as loss function. These two loss functions allow the model to be trained unsupervised. See Figure 5.

The approach of MeanSum is very similar to what the OpinionDigest model does, although it replaces the encoder with a pre-existing tagging model that extracts all the opinion phrases from a given review. This allows the model to filter the selected opinions based on aspect category, where MeanSum can only calculate the mean from the all encoded reviews. It also makes the model more transparent, making it easier for developers and researchers to use and understand. It would theoretically also make the model computationally more expensive, as it can be assumed that extracted opinion phrases constitute a higher dimensional latent space then the final hidden and output states of the LSTM in the encoder of MeanSum.

In a 2021 paper, Amplayo et al. [39] use a reconstruction loss to train an LSTM network, similar to MeanSum, but split the encodings into an aspect and a sentiment specific encoding. These encodings are used to model probability distributions, which are later used to create a synthetic dataset. The model learns the aspect and sentiment embedding matrices by using the rating accompanying each review. In addition to the reconstruction loss function, they add the disentanglement loss function, which is a measure of both how *well* the sentiment probability distribution predicts the rating, as well as how *bad* the aspect probability distribution predicts the rating. Assuming that the rating correctly reflects the sentiment value of each review, this loss function is able to train the sentiment embedding matrix to capture all the sentiment related information, and the aspect embedding matrix to capture all the non-sentiment related information.

Iso et al. [1] observed that most of the summaries produced by the state-of-the-art opinion summarization models are very generic, due to the simple averaging of the latent vectors, representing the encoded input reviews. It is hypothesized that these generic summaries are caused by the L_2-norm of the produced summaries being very low. The L_2-norm, or Euclidean norm, is the length of a vector, and the paper poses that "as we expect each dimension in the latent space to represent a distinct semantics, L_2-norm shrinkage may cause some information loss in the summary vector". To address this problem, they developed a latent vector aggregation framework called COOP, which takes as input the encoded reviews and tries to predict an optimal encoded summary, based on the word overlap between the original review and the predicted decoded summary. See Figure 6.

With this simple new metric, the word overlap, the COOP framework is able to improve upon most of the state-of-the-art opinion summarization models by making the summaries more informative. Since ROUGE is not able to measure the informativeness of summaries, Iso et al. also used human evaluation to measure the performance in addition to the ROUGE score. They found that COOP improved upon existing opinion summarization models, both automatically measured by the ROUGE score as well as with human evaluation regarding the informativeness.

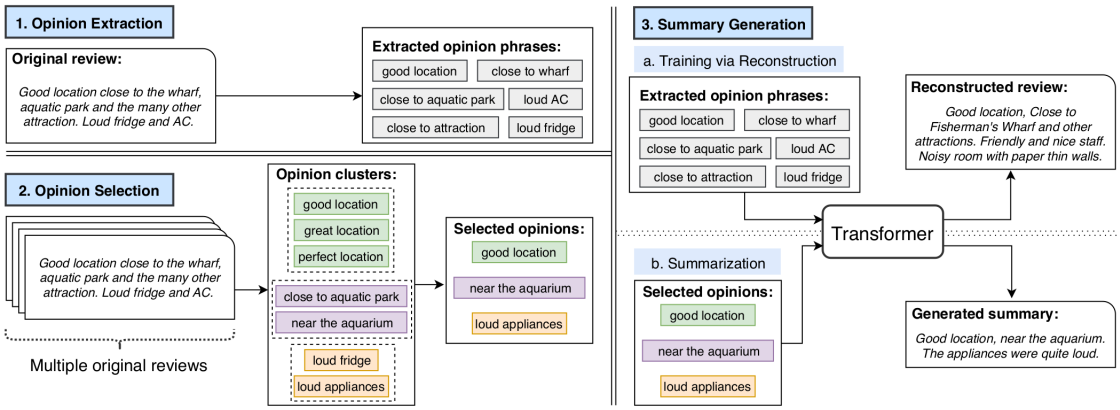

Fig. 4: OpinionDigest framework [37], where (1) opinions are extracted from reviews, (3a) those reviews are reconstructed to train the decoder, (2) the most important opinions are selected and (3b) used to decode into a summary.

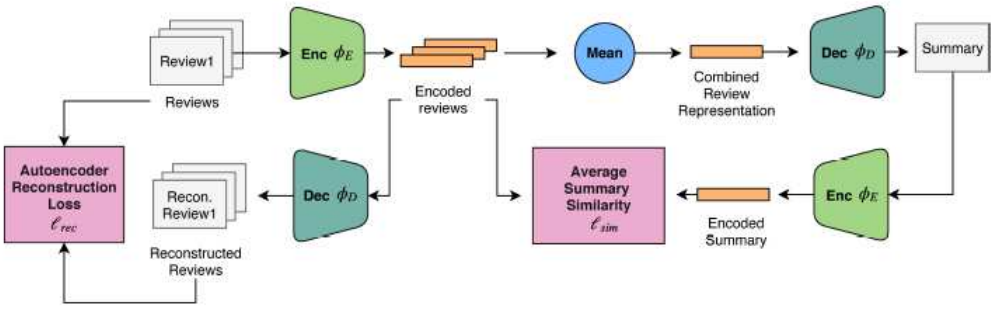

Fig. 5: MeanSum architecture [9], where reviews are encoded to a lower-dimensional latent space, then individually decoded to train the decoder, and also aggregated to be decoded into a summary.

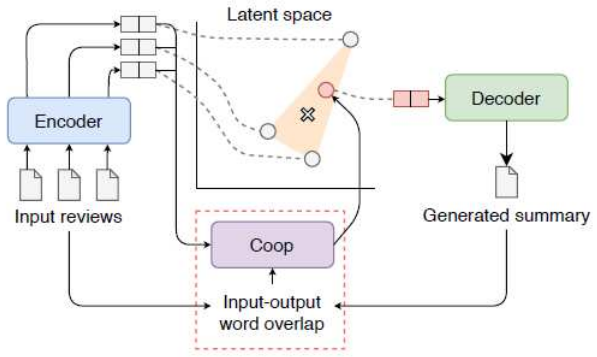

Fig. 6: COOP Framework [1], which improves the MeanSum framework [9] by replacing the mean aggregation with an aggregation based on maximizing input-output word overlap.

3.3 Text Quality

With the Research Question 1 being "How can the informativeness of text be measured", it would also be instructive to look at previous work regarding the measurement of text quality in general. In the field of NLP, the quality of text is measured for a number of reasons. In some cases, the quality of each term is measured individually, for instance with keywords extraction. Here the importance of each word is calculated, which can in turn be used to classify, summarize or cluster documents. Wu et al. [40] present a novel approach to this. They calculate the informativeness of a term t in a certain context c_i by measuring the semantic similarity between the given context and all other contexts found for the term, denoted by $U_f(t)$. More specifically, they use the following equation:

$$I(t, c_i) = \sum_{c_j \epsilon U_f(t)} k(c_i, c_j) \cdot CA(c_j), \tag{4}$$

where

$$CA(c_k) = \sum_{l} DA(d_{kl}). \tag{5}$$

$k(c_i, c_j)$ is the semantic relatedness and $DA(d)$ denotes the authority of a certain document (a Wikipedia page can be given more authority than some blog post for instance). The equation given above can be explained intuitively with the following example. Take the context "PL/SQL is one of three key programming languages embedded in the Oracle Database". The idea is that if you search on the internet for "programming", the results will be more different from the

given context than if you search for "PL/SQL", so the semantic relatedness of the given context and other contexts with that term will be lower, and therefore the term "PL/SQL" is more informative than the term "programming" in that context.

In other cases, the quality of larger volumes of text is considered. Horn et al. [41] attempt to measure how readable or informative a certain document is, with the goal of ranking documents in a search engine or filtering out the least useful documents from a corpus. They do this by calculating the factual density fd of a document d. They use an OpenIE (Open Information Extraction) system called ExtrHech to extract all the facts from a given text, and simply divide the amount of facts by the total amount of text, so:

$$fd(d) = \frac{fc(d)}{size(d)}, \tag{6}$$

where fc denotes the fact count and $size(d)$ denotes the amount of words in document d.

Open Information Extraction here is the extraction of triples, representing facts, which take the form of {*noun*, *verb*, *noun*} ({child, rides, bicycle} for example). ExtrHech takes POS-annotated text, which in the paper is provided by the Freeling-2.2 system, and filters out most of the useless facts. Although a relatively simple system, it is shown that it is a feasible measure of informativeness of text on the Web.

In the paper by Iso et al. [1], discussed in Section 3.2, the quality of summaries of customer reviews is also measured with two metrics, namely text length and information amount. For information amount, they take an approach inspired by the field of information theory as proposed by Shannon [42]. Iso et al. estimate the entropy of a summary by training an RNN on the original dataset to predict the next word in a sentence, and then use the original negative log probability function to calculate the information amount H of summaries X:

$$H(X) = \sum_{i=1}^{n} -log(P(x_i)). \tag{7}$$

Here the result of an opinion summarization model X consists of summaries $\{x_1, \ldots, x_n\}$ and $P(x_i)$ is given by the RNN. This entropy is then used as an indication of how informative a textual summary is.

A paper by Liu et al. [10] has the goal of detecting low-quality customer reviews in order to be able to get the most useful reviews to a retailer or consumer. They train a Support Vector Machine (SVM) to classify reviews as either high quality or low quality, and use human annotators to provide the ground-truth. For informativeness, the SVM uses features such as the number of sentences in the review, the average length of a sentence, the number of products in the review or the number of product features in a review to predict the quality of a review. They find that with only sentence level features, the quality of a review can only be predicted with an accuracy of around 73%, and the word level and product feature level features can increase this percentage to around 83%, while the readability and subjectiveness features contribute next to nothing.

Antiqueira et al. [43] modeled text as a graph, where each term represents a node, and a graph was constructed by defining an edge between two nodes when the two associated words are adjacent. They found that the coherence and cohesion of the text, as evaluated by human judges, was negatively associated with the out-degrees (i.e. the amount of edges going from a node), clustering coefficient (i.e. the degree to which groups of nodes tend to cluster together) and deviation from a linear network growth of the graph (i.e. if the network grows linearly as each new word is added to the network).

Research done by Mesgar et al. [44] takes a more complex approach. A neural network is developed to predict text quality in two domains, namely readability and essay scoring. For readability assessment, a number of English texts are taken from the British National Corpus and from Wikipedia, and the network needs to predict how readable they are, as evaluated by human judges. In the case of essay scoring, a dataset is taken from the Automated Student Assessment Prize competition from Kaggle[1], and the network needs to predict which student essay ranks highest, again as evaluated by human judges. To accomplish this, the researchers develop a local coherence model. Specifically, they use an LSTM to encode the semantic information between (the most similar words of) two adjacent sentences, and a CNN that subsequently encodes the semantic information across all the sentences in a text. To evaluate their performance, they compare their model with the open-source essay scoring system EASE, and fail to improve upon the QWK score, which is used to measure the agreement between the generated ranking and the human ranking. However, by combining their local coherence model with EASE, they do manage to improve upon the performance of EASE, with a QWK score of 0.728 as compared to the 0.705 of EASE.

4 Methodology

This sections consists of two parts. First, new metrics are proposed by which to measure "informativeness". Second, a new model is proposed that could improve performance on these metrics.

4.1 Metrics

With regards to the first research question, "How can the informativeness of text be measured", this paper takes the liberty of answering it by proposing the following metrics, inspired by other papers: the information amount, the relevance and the frequency of common words.

4.1.1 Information Amount

As mentioned in Section 3.2 and 3.3, Iso et al. [1] used the information amount as an indication of informativeness to the reader, where information amount is defined by

$$H(X) = \sum_{i=1}^{n} -log(P(x_i)). \tag{8}$$

Here the result of an opinion summarization model X consists of summaries $\{x_1, \ldots, x_n\}$ and the probability of each summary is given by a trained RNN. The intuition

1. https://www.kaggle.com/competitions/asap-aes/overview

behind this is that the more predictable a summary is, the less information it must contain. In this paper, we will use this idea as well, albeit in the following, simplified form. We will use the `t5-small-next-word-generator-qoogle` from the HuggingFace Hub [12], a generic English next-word-predictor, to measure the predictability of a summary. To clearly communicate the results of this metric, the ratio of correctly predicted words will be presented.

4.1.2 Relevance

A problem which could be observed in the results of the work by Iso et al. [1] is the inclusion of more irrelevant information in summaries during the pursuit of more informative summaries. To address this problem, we will try to measure the relevance of a summary. This is done by using the general subject of a summary, i.e. the `product_category` in the Amazon dataset and the `categories` in the Yelp dataset, and calculate average distance between the words used in the summary and the subject. Since word vectors are grounded in semantic similarity, they can be used to calculate the semantic distance between two words. More specifically, the average Euclidean distance of the words in the summary to the product category name is taken, where the Euclidean distance between two n-dimensional vectors is defined by

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n + q_n)^2}. \tag{9}$$

The word vectors from the pre-trained Word2Vec model `word2vec-google-news-300` from Gensim [45] will be used, where each word vector has 300 dimensions, and each value is normalized between -1 and 1. This means that the lower bound for this metric is given by the distance of two identical vectors, which will be zero since

$$\begin{aligned} d(p,p) &= \sqrt{(p_1 - p_1)^2 + (p_2 - p_2)^2 + \cdots + (p_{300} - p_{300})^2} \\ &= \sqrt{0^2 + 0^2 + \cdots + 0^2} = 0. \end{aligned} \tag{10}$$

The upper bound for this metric is given by the distance of two vectors who have the largest difference in value on each dimension, which is 2, since each value is between -1 and 1. This gives

$$\begin{aligned} d(p,q) &= \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_{300} + q_{300})^2} \\ &= \sqrt{4 + 4 + \cdots + 4} \\ &= \sqrt{300 * 4} = \sqrt{1200} = 20\sqrt{3}. \end{aligned} \tag{11}$$

So the upper bound for this metric is $20\sqrt{3}$.

4.1.3 Word Frequency

The last metric that will be taken into account is the frequency of common words. Inspired by the approach of Wu et al. [40] (see section 3.3), the informativeness of a certain term will also be measured by how often it occurs in other contexts. Although Wu et al. take a more sophisticated approach, where they calculate the semantic similarity between the given context and all other contexts the term occurs in, here it would be too computationally expensive to take into account all the other contexts for each word. Instead, the most common English words are taken from the Corpus of Contemporary American English [46], a corpus containing texts from sources such as TV and movie subtitles, academic texts, newspapers, blogs, works of fiction, etc. Since the explicit goal of this corpus is to be genre-balanced, the most common words of the corpus are assumed to be relatively uninformative, as they appear in a lot of different contexts. The validity of the use of this metric is substantiated by the prevalence of TF/IDF in NLP research, as this metric rests on the same assumption. The importance of a term in a document is calculated by multiplying with the Inverse Document Frequency, which depends on how often the term occurs in other contexts. The more it occurs in other contexts, the lower the IDF, and the lower the importance, or informativeness.

The top 5000 most common words from the COCA dataset are freely available, and those are the words that will be used for this metric. Each word in the dataset has a `freq` attribute, which gives the frequency of the word occurring in the corpus. This value ranges from approximately 50 million for the word "the", to 11 thousand for the word "bizarre". To give an indication for the informativeness of a summary, these word frequencies are summed up for all the words in the summary that are also in this dataset, and divided by the total amount of words a summary contains. This results in the following aggregate.

$$a(W) = \frac{\sum_{w_i \epsilon W} freq(w_i)[w_i \epsilon COCA]}{|W|}, \tag{12}$$

where $freq(w_i)$ gives the frequency of word w_i in the COCA dataset, and summary W consists of words w_i.

4.2 Model Proposal

In this subsection, the second research question, "How can the informativeness of existing opinion summarization methods be improved?", will be answered by proposing a model that will attempt to improve the informativeness of existing opinion summarization methods with a combination of topic modeling and sentiment analysis, as shown in Figure 7. This model will be called TopSum [2], because it is a summarization model that bases its summaries on topic associations. It is used to generate one summary for one product, based on eight customer reviews for that product. It roughly consists of four steps, namely sentence extraction, topic modeling, ranking per topic and text summarization.

4.2.1 Sentence Extraction and Topic Discovery

The first step of TopSum is to extract the sentences from all the reviews of a certain product. This is done by splitting each review text on the '.' character. This approach is relatively simple as compared to the phrase extraction from the OpinionDigest model [37]. Although many regular keyphrase extraction systems exist [47] [48] [49], using a reliable opinion phrase extraction system would be more complex. Simple sentence extraction is used for TopSum, both to reduce the dependency on external libraries, and to keep the model transparent and understandable.

The second step of TopSum is a more central part of the research, and involves topic modeling of the extracted sentences. For reasons discussed in section 2.2.2, BERTopic

2. https://github.com/EricvanSchaik/TopSumProject.

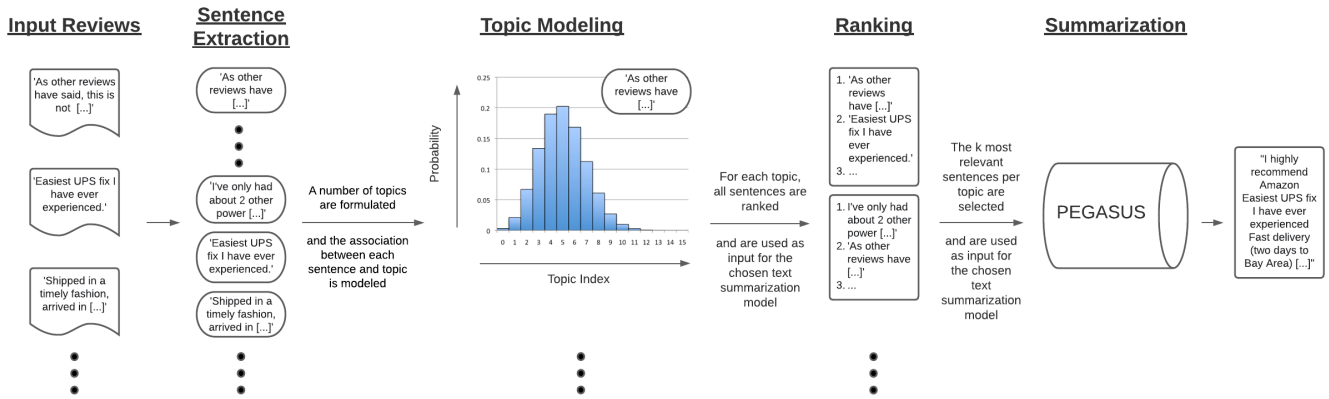

Fig. 7: Given a number of input reviews for one product or service, the TopSum model extracts the sentences, performs topic modeling to retrieve the most relevant aspects, gives a ranking of sentences per topic, and summarizes the 20% most important sentences per topic. This produces one summary for the given product or service.

[31] is chosen for this. BERTopic is initialized with 10 topics, and is trained on the entire dataset. A topic model per product or service was considered as well, however with only 8 reviews per product, this does not result in a stable topic model. The results of this step are a topic model, which can be used to retrieve the 10 topics or to assign topic probabilities to new text, and for each review a probability distribution over those topics (i.e. what is the probability that a review is about a certain topic).

4.2.2 Ranking per Topic

The third and most complex step of TopSum is to rank the sentences per topic, in order to determine which are the most important sentences for a certain topic. This ranking will be based on three aspects, namely the topic association, the sentiment deviation and the L_2 norm. The topic model from the first step is used to produce a topic probability distribution, this time on a per-sentence basis, resulting in a value between 0 and 1 per topic per sentence.

Ideally, a summary of customer reviews would include multiple aspects of the product or service, described with opinion phrases that reflect the average sentiment polarity of the customer reviews with respect to those aspects. This model attempts to do this by attaching more importance to sentences in the customer reviews that are closer to the average sentiment value of a topic, during this ranking phase. To achieve this, first the average sentiment value per topic needs to be calculated. This is done as follows.

$$\bar{s_i} = \frac{\sum_{j=0}^{n}(w_{ij} \cdot c_j)}{\sum_{j=0}^{n} w_{ij}}, \tag{13}$$

where $\bar{s_i}$ is the average sentiment value for topic i, n is the amount of reviews, w_{ij} is the topic association between topic i and review j as predicted in the second step, and c_j is the compound score of review j, given by the VADER dictionary (see Section 2.2.3). After having calculated the average sentiment value per topic, the sentiment deviation per topic per sentence is calculated by taking the absolute difference between the average and the compound score of the sentence.

Finally, the L_2 norm is calculated. The L_2 norm, or Euclidean norm, is the Euclidean distance between a vector and the origin, and is given by the following equation.

$$||\boldsymbol{x}||_2 = \sqrt{x_1^2 + \cdots + x_n^2}. \tag{14}$$

Here n is the dimension of the vector and x_k is the k-th dimension of vector x.

The L_2 norm is calculated for each word in a sentence, and the average taken. Words not in the pre-trained Word2Vec [28] dictionary are ignored.

With these three aspects, a final score is calculated per sentence per topic, as follows.

$$f_{ij} = p_{ij} + \alpha \cdot -d_{ij} + \beta \cdot l_j. \tag{15}$$

Here, α and β are two hyperparameters of this model, and are chosen to be 0.1 and 0.05 after experimentation. f_{ij} is the final score of topic i and sentence j, p_{ij} is the topic prediction, d_{ij} is the sentiment deviation and l_j is the Euclidean norm.

The inclusion of the Euclidean norm in the calculation of the final score is inspired by the results of Iso et al. [1], that show that there is a correlation between the L_2 norm of latent vectors and the information amount of the summaries. Here the situation is somewhat different, as the L_2 norm of the word vectors according to Word2Vec [28] are taken, while the latent vectors of the paper by Iso et al. are the embeddings given by the encoder of the transformer architecture. Therefore, this step makes the additional assumption that the correlation would still hold between the Euclidean norm of a Word2Vec vector and the semantic information of that word.

On a surface level, this assumption is reasonable since a vector that has a larger L_2 norm (i.e. that is farther away from the origin of the vector space) would require more bits to encode. Therefore, Word2Vec would be a highly inefficient encoding scheme if the most common words would be furthest away from origin. More specifically, this assumption is also reasonable considering the main contribution of Word2Vec. Word2Vec provides word vectors where the cosine similarity indicates the semantic similarity

(i.e. words that are semantically similar have word vectors that are close together in the vector space), and it should be expected that common words have more words that are semantically similar than very specific uncommon words. Assuming the origin is the average of all the word vectors, most word vectors will be in that area, so the most common words should be as well, giving them a small Euclidean norm.

In addition to these theoretical arguments, this choice will also substantiated by the calculation of the Euclidean norm of the 5000 most common English words, see Section 6.1.2

4.2.3 Text Summarization

Within each topic cluster, sentences are now ranked based on relevance, sentiment value and information. The last step of this model is to summarize these sentences, and this is done with the `pegasus-cnn_dailymail` model. The 20% most important sentences from each topic cluster are selected and used as input for PEGASUS. This percentage is another one of the hyperparameters for this model, initially chosen somewhat arbitrarily and therefore open to experimentation.

An example of the whole process is given in the form of Figure 7. Three of the eight input reviews are shown, concerning an Amazon product called "APC UPS Battery Replacement". After cutting the reviews up into sentences, BERTopic is used to calculate ten different topic models, along with a probability distribution over the topics for each sentence. Only one of these distributions is shown for simplicity. In the next step, all the sentences are ranked for each topic, to reflect how important each sentence is in describing the average sentiment for each product aspect. In the last step, only the 20% most important sentences per topic are fed into the PEGASUS model to produce a summary.

5 Experimental Setup

5.1 Datasets

The two most popular datasets in opinion summarization research are used for the experiments, namely the Yelp dataset and the Amazon dataset, both also used by Iso et al. [1] and available at https://www.yelp.com/dataset and http://jmcauley.ucsd.edu/data/amazon/links.html, or via the HuggingFace website https://huggingface.co/datasets. For both datasets, only products are chosen for which there are exactly 8 reviews, following the example of the MeanSum paper [9]. With this condition, 10.000 reviews are sampled from the Amazon Electronics dataset and 8960 reviews are sampled from the Yelp dataset. For Amazon, calculating the relevance is straightforward, since it can be assumed that the general subject of each summary is "Electronics". For the Yelp dataset, this requires some preprocessing, since an example of the `categories` feature is "Food, Ethnic Food, Nightlife, Restaurants, Dive Bars, Bars, Vietnamese, Specialty Food, Street Vendors, Event Planning & Services, Asian Fusion, Caterers, Food Stands". Here, simply the first word of all the categories is taken as the subject of the summary.

5.2 Baseline Models

The frameworks that are used as baselines are the text summarization PEGASUS [2] and the opinion summarization model COOP [1], both as explained in Section 3.

On the HuggingFace Hub [12], Google has published a number of versions of PEGASUS, most notably `pegasus-xsum`, `pegasus-cnn_dailymail` and `pegasus-large`, where the first model is trained on the XSum dataset, the second model is trained on the CNN/DailyMail dataset, and the last model is the original model from the paper, trained on a wide range of datasets, including XSum and CNN/DailyMail. As mentioned in Section 3.2, the most significant difference between the XSum and the CNN/DailyMail dataset is the length of the summaries, as the XSum summaries are restricted to just one sentence. For the purpose of summarizing customer reviews, it would be useful to include multiple aspects in the summaries, as the associated sentiment per aspect can vary, and each reader could be interested in a different aspect. Therefore, summaries longer than one sentence would be preferred, so the `pegasus-cnn_dailymail` model is chosen for the following experiments.

The COOP framework is taken from the GitHub repository [3], and this model is chosen as a baseline for two reasons. It is based on and very similar to the MeanSum model [9], which is an influential framework in opinion summarization, and is therefore being used as a baseline in many other opinion summarization papers as well [50] [51] [51] [37] [52] [53]. Also, it is particularly relevant to this research as it is one of the few papers that also sees the low amount of information as one of the key problems in new unsupervised opinion summarization frameworks, and some ideas to solve this problem serve as inspiration for the proposals laid out in this paper.

5.3 Human Evaluation

A survey has been set up to test which model produces more informative summaries. The survey contained 12 questions in total, 6 for each dataset. Each question focused on one product, first giving the full texts of the 8 reviews and 3 summaries (from our proposed TopSum, PEGASUS [2] and COOP [1]), followed by asking how they would rank the summaries from most to least informative. An example of such a question is given in Appendix B.

This survey was distributed in two ways. It was distributed to personal contacts (students at the University of Twente for instance), and on the SurveySwap [54] platform. On the SurveySwap platform, students can swap their surveys, so a user has to fill in surveys of other students in order to get participants for his survey. The source of the survey responses will be given when presenting the results of this human evaluation.

After collecting the responses from the survey, the results are processed to be able to measure a difference in performance. To this end, a score is calculated for each model, on each dataset and for each survey respondent source. This score is calculated as follows. A response is in the form of a ranking of the models from most to least informative.

3. https://github.com/megagonlabs/coop

For each individual response, the most informative model is given two points, the second most informative model is given one point and the last model is given zero points. All these points are summed up for each respondent for each question, giving each model a score between 0 and 12 per dataset per participant.

6 Results

This section is divided into three subsections. First, the results of the intermediate experiments are presented, which were used during the process of designing TopSum to support various design decisions. Second, the performance of the proposed model as compared to the baseline models is considered. This can be considered the main part of this section. It is followed up by the results of the human evaluation.

6.1 Intermediate Experiments

6.1.1 Sentiment Deviation per Sentence

In order to decide whether to split the reviews into sentences before performing ranking and summarization, an experiment was carried out. It was assumed that one review was likely to cover multiple aspects, and that the reviewer would feel differently about each aspect. Since the proposed model tries to select the most important pieces of text per topic, a smaller level of granularity could be more appropriate, and the sentence would be an obvious candidate, due to the ease of splitting text into sentences. To test this assumption, the deviation of sentiment of individual sentences with respect to the whole review was measured and visualized. A larger deviation would suggest a more meaningful distinction between individual sentences, and would support the decision of splitting the reviews into individual sentences before summarization. The results of this experiment are show in Figure 8. For this experiment, 200 reviews were sampled from the Amazon dataset in such a manner as to reflect the average sentiment value distribution of the full dataset, and sorted based on sentiment value. The blue line in Figure 8 shows the average sentiment value of the 200 reviews, and the red line reflects the sentiment value of the individual sentences of the reviews. As shown in the figure, the sentiment of individual sentences roughly correlates with the sentiment of the reviews, as expected, although large deviations are visible as well. Especially in reviews with a compound score between between 0 and 0.5, the deviation is substantial. The result of this experiment seems to support the decision to split the reviews into sentences.

6.1.2 Information in Most Common Words

To further substantiate the claim that the L2-norm of Word2Vec vectors negatively correlates with how common a word is, another experiment was carried out. Again, COCA [46] was used to get the most common English words, and the L2-norm of each word was calculated. More specifically, the 2000 most common words were taken from the corpus, sorted on frequency, and the average L2-norm of 10 subsequent words was calculated and plotted. The result is in Figure 9, where the words with COCA index 0 are the 10 words that have the lowest frequency in the corpus (of the 2000 most common words) and a higher COCA index signifies a more common English word. With this word selection, it is shown that there is a strong negative correlation between word frequency and L2-norm, and that therefore ranking based on L2-norms should produce more informative summaries.

6.2 Performance of Model

The final results for the Amazon dataset are shown in Table 1a and for the Yelp dataset in Table 1b. As shown, our TopSum model performs well in terms of predictability (i.e. information estimation), with a score of 0.184 on the Amazon dataset and a score of 0.177 on the Yelp dataset. This is close to the scores of PEGASUS of 0.180 and 184, and both are much better scores than those of COOP, which are 0.314 and 0.297, for the Amazon and Yelp dataset respectively. Note that the numbers reflect the ratio of well predicted words, so a lower score indicates a higher information estimation.

The model performs worse than the baseline models in terms of relevance, with a score of 3.458 on Amazon and a score of 3.481 on Yelp as opposed to 3.420 and 3.422 for PEGASUS and 3.323 and 3.366 for COOP on the Amazon and Yelp datasets respectively. Since the relevance metric gives the average distance of the summary to the subject, a lower distance means it is more relevant.

TopSum performs significantly better in terms of common word usage, with a score of $4.363 \cdot 10^6$ on Amazon and a score of $4.128 \cdot 10^6$ on Yelp, where PEGASUS reached scores of $4.634 \cdot 10^6$ and $4.744 \cdot 10^6$ and COOP got scores of $6.557 \cdot 10^6$ and $6.768 \cdot 10^6$ on Amazon and Yelp respectively.

TopSum also performed significantly better in terms of sentiment deviation, with a score of 0.432 on Amazon and a score of 0.372 on Yelp, while PEGASUS got 0.547 and 0.536 and COOP 0.477 and 0.457 on Amazon and Yelp respectively. It must be noted that sentiment deviation is the only metric that is already taken into account during the ranking and filtering part of the model, and therefore is not a good measure of how well our model is performing. However, it is so central to the difference between text summarization and opinion summarization that it is nevertheless shown here.

An example summary for each dataset and for each model is given in Appendix A.

6.3 Survey Results

For the human evaluation, 13 participants were used, 7 of which were from personally contacted, and 6 of which were gathered from SurveySwap. The results of the survey are shown in Figure 10.

The results are presented as a bar chart, where each bar represents the performance of one model on one of the datasets. Each bar is divided into two parts, where the bottom darkly-colored part shows the results from the personal contacts, and the top lightly-colored part shows the results from the SurveySwap participants. To illustrate the performance of each model, a score has been calculated for each bar, and is shown on the y-axis. The exact score of each case is shown in the separate bars as well, with the total score of each model on each dataset being shown on

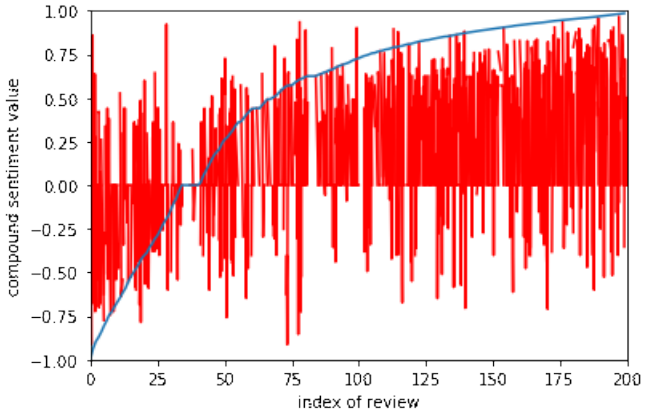

Fig. 8: Sentiment deviation of sentences of 200 Amazon reviews

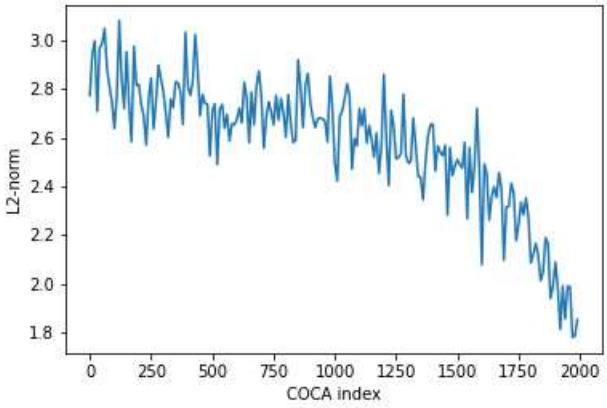

Fig. 9: Correlation of frequency of top 2000 most common words of COCA [46] and L2-norm

Summarization Model	Predictability	Relevance	Word Frequency	Sentiment Deviation
TopSum	0.184	3.458	$4.363 \cdot 10^6$	0.432
PEGASUS [2]	0.180	3.420	$4.634 \cdot 10^6$	0.547
COOP [1]	0.314	3.323	$6.557 \cdot 10^6$	0.477

(a) Amazon

Summarization Model	Predictability	Relevance	Word Frequency	Sentiment Deviation
TopSum	0.177	3.481	$4.128 \cdot 10^6$	0.372
PEGASUS [2]	0.184	3.422	$4.744 \cdot 10^6$	0.536
COOP [1]	0.297	3.366	$6.768 \cdot 10^6$	0.457

(b) Yelp

TABLE 1: Automatic evaluation of summarization models by proposed metrics

top of the bars. Because each model has a score between 0 and 12 per dataset per participant (see Section 5.3), in total each model has a score between 0 and 84 points per dataset for the personally gathered participants and between 0 and 72 per dataset for the SurveySwap participants.

As shown in Figure 10, the perceived informativeness of TopSum on the Amazon is close to the perceived informativeness of PEGASUS, with the regular respondents rating the informativeness of TopSum even higher. COOP performs worst on the Amazon dataset on every aspect, although the difference becomes small when looking at the regular respondents.

Overall, the results from the regular respondents roughly correspond with the results from the respondents from SurveySwap. One noticeable outlier would be the perceived informativeness of the COOP model on Amazon, which the regular respondents perceive as much more informative than the respondents from SurveySwap.

Although the COOP and PEGASUS model perform similar on Yelp as their Amazon counterpart, the perceived informativeness of the TopSum model on the Yelp dataset is considerably worse, both when considering the respondents from regular sources as well as those from SurveySwap. This trend is similar to that of the relevance score considered in

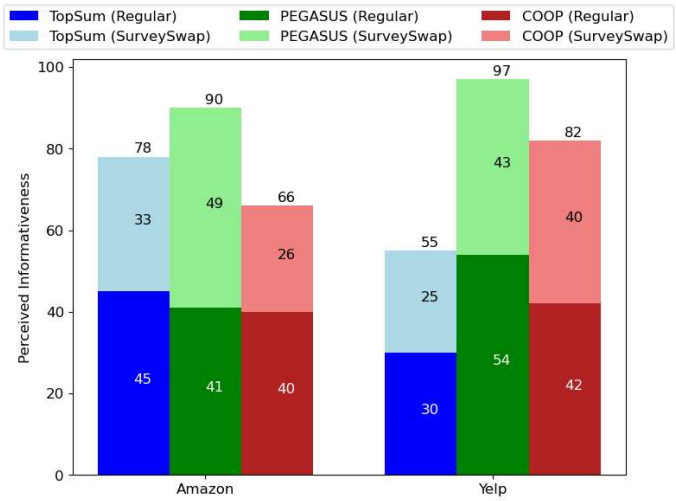

Fig. 10: Human evaluation of perceived informativeness of different summarization models on two datasets, as measured by surveys

Section 6.2.

7 Discussion

In this section, the results are interpreted in terms of the implicit hypothesis, the research questions are answered, some limitations are highlighted and suggestions for future work are suggested.

7.1 Hypothesis

The results of the experiments do not fully support the hypothesis that by ranking and filtering the most important sentences before using text summarization, the gap in text quality between text summarization and opinion summarization can be reduced. Especially in the case of our measure of relevance, the model underperforms. This could be because of the topic modeling part of the model, which is dependent on external libraries (BERTopic [31] in this case). Producing an accurate number of topics that reflect aspects of a product/service given a large number of documents is a hard problem. Inaccurate topics may well be produced with low-quality of input text even with BERTopic, which is the state-of-the-art in the field. Another possible reason for the reduction in relevance is the broadness of the dataset. With the Amazon dataset, reviews were taken from the product category "Electronics", and that category includes a wide range of products such as cables, TV's and headphones. Since the category is so broad, it seems likely that many topics of the resulting topic model are specific to a type of product within the category "Electronics". This would mean that when ranking the sentences based on association with a product aspect, all aspects are taken into account equally, although only a small subset of the aspects are relevant. This would result in sentences with low topic associations still being selected as the most important sentences for a product, and therefore resulting in a lower relevance score. The significant difference between the Yelp dataset and the Amazon dataset supports this analysis. With Yelp, businesses do not belong to one category. Yelp uses the categories more as tags, where they attach multiple categories to one business, so that there is potentially partial overlap between the categories of two businesses. This makes it more complex to train a topic model on one category, and therefore the topic model used here is trained on the entire Yelp dataset. It is likely that this has lead to even more distinct topics, where most reviews have little to do with most topics. Therefore, the relevance score of TopSum is significantly worse on the Yelp dataset (3.481) than on the Amazon dataset (3.458).

The small-scale human evaluation further supports the notion that the topic model could be partly responsible for the suboptimal performance. On the Amazon dataset, the TopSum model performs around as good as PEGASUS, while on Yelp it is the worst performing model. Although this is reminiscent of the difference in relevance scores mentioned above, the Amazon and Yelp datasets differ in a number of ways, and many other reasons for the difference in performance are imaginable.

If the suboptimal performance is due to the topic model, it would be one of the accepted risks of depending on an external topic modeling framework. The problem might partly be addressed by tweaking the topic model and adding some preprocessing steps to the TopSum model. For instance, the topic model could be trained only on one product, so as to make sure that all the aspects are specific to that product. This was already explored during experimentation, and it was rejected for this project, primarily for computational and pragmatic reasons. Since BERTopic needs to fine-tune BERT, a relatively large number of documents are needed in order to produce a stable topic model. In the Amazon product category "Electronics", the largest amount of reviews for one product was around 8000 reviews, and that seemed to be enough for a good topic model. However, the rest of the model proved to be so complex that these large volumes of text per summary would be too computationally expensive. Also, more pragmatically, the amount of products with this number of reviews is so small, that a summarization model that would only work under these specific circumstances would not be useful.

Another way to improve the topic model could be to filter out some type of words, such as stop words, pronouns or words with a strong sentiment value (since producing text with the correct sentiment value is already controlled for by other parts of the model). Also, the semantic similarity between a topic and the category could be taken into account, so that topics that are more semantically similar to the category are preferred. All this could produce more reliable topic models, and therefore higher relevance scores.

7.2 Research Questions

Considering the first research question, "How can the informativeness be measured", this paper has offered some tentative metrics by which an indication of informativeness could be given. Significant differences between the performance of the summarization models with these new metrics conform to intuitions regarding the text quality differences, supporting the claim that these metrics give an indication of informativeness.

An attempt to answer the second research question, "How can the informativeness of existing opinion summarization methods be improved?", has resulted in a new model proposal which ranks and filters review sentences, before performing text summarization on them. The third research question, "How can the sentiment value be maintained when focusing on informativeness?", is answered implicitly in this model, as the sentiment value of the sentences is taken into account during the ranking. This approach can also be considered as a combination of extractive and abstractive summarization, where all the sentences are taken into account for the correct sentiment orientation, however only a subset of the input text is used for the production of high-quality text. Although this has not improved opinion summarization on all fronts, the significant differences between it and the baseline models suggest a meaningful distinction between the proposed architecture and state-of-the-art text/opinion summarization models. Therefore, this new model could be a viable starting point for further research.

7.3 Limitations

In addition to the limitations of the topic model mentioned above, a number of other limitations on this research should be considered. The main limitation of this research is the arbitrariness of the proposed metrics. Although partially inspired and substantiated by previous research, the choice of these metrics is also based on subjective intuition regarding the quality of the output of existing opinion summarization models. This was considered necessary, as the perceived performance of existing opinion summarization models and their ROUGE measurements seemed to correspond poorly. Due to the simplicity, popularity and longevity of the usage of ROUGE, it could be considered a more objective measure of summarization performance, although this paper poses that an alternative could be useful. Therefore, this limitation is inherent in the conceptualization of this research.

Another limitation is that more computational resources would have been desirable. Due to the usage of the large topic model, the state-of-the-art abstractive text summarization model and the complex per-sentence ranking calculations, running the full TopSum model was relatively expensive. This prohibited extensive experimentation with hyperparameters such as those mentioned in Section 4.2.2 and 4.2.3, with different amounts of reviews per summary (only the case of 8 reviews per summary is considered in this research, in line with the experimental setup of similar papers), or more types of text summarization models. The scarcity of computational resources seems to be a common limitation in NLP research. Starting with the introduction of BERT by Google [4], the largest advances in NLP research have been made by large companies, having the resources available to train very complex models with vast amounts of data. To fine-tune a model for more specific tasks often entails using the pre-trained language models and training some extra layers with task-specific labeled data, making the model even more complex and making computational resources scarce. The more consumer-centric approach of this paper does not include fine-tuning with labeled data, instead it combines a number of language models and NLP techniques to produce better summaries, and its complexity is therefore a sum of the complexity of the different language models and techniques. In this case, the state-of-the-art is used as components of the TopSum model, and therefore the extend of this limitation could have been reduced by choosing simpler components. Some experimentation in this direction has shown worse intermediate results, and therefore has not been explored further.

Since the design of TopSum is based partly on intuitive choices, these choices could have been founded with more research as well. For instance, in section 6.1.1, the choice of cutting text into sentences before processing is substantiated by showing the sentiment deviation of individual sentences with respect to whole reviews. This could be expanded by showing the deviation of topic associations of sentences, or the differences in L2-norms per sentence. Additionally, to support the design choice of using topic modeling to improve relevance, some experimentation regarding the relevance of those topics could be instructive as well.

7.4 Future Work

The limitations mentioned above naturally lead to a number of opportunities for future work. For instance, more experimentation could be done with more computational resources, or by making the model more efficient. These experiments could entail tweaking parts of the model, or could address certain design choices. Also, the arbitrariness of the metrics could be addressed by looking for other metrics, so that a fuller picture of informativeness can be given to verify whether the metrics correspond to perceived informativeness in text.

In addition to simply addressing the limitations mentioned above, some more opportunities for future work could be mentioned. One important direction could be adding the option to generate aspect-specific summaries. This option seemed to be a priority in opinion summarization studies at some point in the past, however since the introduction of transformers [3] and MeanSum [9], aspect-based summarization is somewhat neglected. This model has been designed with aspect-based summarization in mind, the effectiveness of which largely depends on the effectiveness of the topic modeling part of the TopSum model. Therefore, any future work in this direction should consider improving the use of topic modeling in TopSum, if it is shown to be necessary.

8 Conclusion

In this paper, a new approach to opinion summarization has been proposed, addressing the apparent gap in text quality between the output of state-of-the-art text summarization models and opinion summarization models. It starts with the proposal of a number of new metrics, which attempt to gauge the performance of opinion summarization in a more consumer-centric manner, namely by measuring the informativeness of text. These metrics are predictability, relevance and common word frequency, and this paper shows how a new opinion summarization framework TopSum could improve performance on these metrics. This new framework could be considered a combination of extractive

and abstractive summarization, since it selects a fraction of the input text it considers most important, and performs abstractive summarization on it. It is shown that the TopSum model partly bridges the gap between text and opinion summarization, although not on the relevance metric. This could be attributed to the nature of the available datasets, and the resulting topic modeling performance. To conclude, although the TopSum model is not successful on all fronts, it could be considered a viable starting points for research into more useful opinion summarization models.

REFERENCES

[1] H. Iso, X. Wang, Y. Suhara, S. Angelidis, and W.-C. Tan, "Convex aggregation for opinion summarization," *arXiv preprint arXiv:2104.01371*, 2021.

[2] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 328–11 339.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," *ArXiv*, vol. abs/1808.08745, 2018.

[6] A. Cohan, F. Dernoncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, and N. Goharian, "A discourse-aware attention model for abstractive summarization of long documents," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 615–621. [Online]. Available: https://aclanthology.org/N18-2097

[7] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, 2016, pp. 507–517.

[8] Y. Cui, "An evaluation of yelp dataset," 2015. [Online]. Available: https://arxiv.org/abs/1512.06915

[9] E. Chu and P. Liu, "Meansum: a neural model for unsupervised multi-document abstractive summarization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1223–1232.

[10] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang, and M. Zhou, "Low-quality product review detection in opinion summarization," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 334–342.

[11] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1073–1083. [Online]. Available: https://www.aclweb.org/anthology/P17-1099

[12] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] Wikipedia, "Artificial neural network — Wikipedia, the free encyclopedia," http://en.wikipedia.org/w/index.php?title=Artificial%20neural%20network&oldid=1109022706, 2022, [Online; accessed 09-September-2022].

[16] A. Brahme, *Comprehensive biomedical physics*. Newnes, 2014.

[17] P. J. Bickel and K. A. Doksum, *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. Chapman and Hall/CRC, 2015.

[18] Wikipedia, "Recurrent neural network — Wikipedia, the free encyclopedia," http://en.wikipedia.org/w/index.php?title=Recurrent%20neural%20network&oldid=1109264340, 2022, [Online; accessed 09-September-2022].

[19] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

[20] M. O. Topal, A. Bas, and I. van Heerden, "Exploring transformers in natural language generation: Gpt, bert, and xlnet," *arXiv preprint arXiv:2102.08036*, 2021.

[21] E. Filatova and V. Hatzivassiloglou, "Event-based extractive summarization," 2004.

[22] "Text summarization," https://devopedia.org/text-summarization, Feb. 2020, accessed: 2022-9-12.

[23] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, "Text summarization techniques: a brief survey," *arXiv preprint arXiv:1707.02268*, 2017.

[24] E. Sharma, C. Li, and L. Wang, "BIGPATENT: A large-scale dataset for abstractive and coherent summarization," *CoRR*, vol. abs/1906.03741, 2019. [Online]. Available: http://arxiv.org/abs/1906.03741

[25] A. Kornilova and V. Eidelman, "Billsum: A corpus for automatic summarization of us legislation," 2019.

[26] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[27] H. D. Kim, K. Ganesan, P. Sondhi, and C. Zhai, "Comprehensive review of opinion summarization," 2011.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[29] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[30] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[31] M. Grootendorst, "Bertopic: Neural topic modeling with a class-based tf-idf procedure," *arXiv preprint arXiv:2203.05794*, 2022.

[32] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[33] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," 2020.

[34] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2018. [Online]. Available: https://arxiv.org/abs/1802.03426

[35] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, mar 2017. [Online]. Available: https://doi.org/10.21105%2Fjoss.00205

[36] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the international AAAI conference on web and social media*, vol. 8, no. 1, 2014, pp. 216–225.

[37] Y. Suhara, X. Wang, S. Angelidis, and W.-C. Tan, "Opiniondigest: A simple framework for opinion summarization," *arXiv preprint arXiv:2005.01901*, 2020.

[38] Z. Miao, Y. Li, X. Wang, and W.-C. Tan, "Snippext: Semi-supervised opinion mining with augmented data," in *Proceedings of The Web Conference 2020*, 2020, pp. 617–628.

[39] R. K. Amplayo, S. Angelidis, and M. Lapata, "Unsupervised opinion summarization with content planning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, 2021, pp. 12 489–12 497.

[40] Z. Wu and C. L. Giles, "Measuring term informativeness in context," in *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: human language technologies*, 2013, pp. 259–269.

[41] C. Horn, A. Zhila, A. Gelbukh, R. Kern, and E. Lex, "Using factual density to measure informativeness of web documents," in

Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013), 2013, pp. 227–238.

[42] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[43] L. Antiqueira, M. d. G. V. Nunes, O. Oliveira Jr, and L. d. F. Costa, "Strong correlations between text quality and complex networks features," *Physica A: Statistical Mechanics and its Applications*, vol. 373, pp. 811–820, 2007.

[44] M. Mesgar and M. Strube, "A neural local coherence model for text quality assessment," in *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018, pp. 4328–4339.

[45] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[46] M. Davies, "Corpus of Contemporary American English (COCA)," 2015. [Online]. Available: https://doi.org/10.7910/DVN/AMUDUW

[47] T. Schopf, S. Klimek, and F. Matthes, "PatternRank: Leveraging pretrained language models and part of speech for unsupervised keyphrase extraction," in *Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. SCITEPRESS - Science and Technology Publications, 2022. [Online]. Available: https://doi.org/10.5220%2F0011546600003335

[48] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," *arXiv preprint arXiv:1704.06879*, 2017.

[49] K. Bennani-Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi, "Simple unsupervised keyphrase extraction using sentence embeddings," *arXiv preprint arXiv:1801.04470*, 2018.

[50] R. K. Amplayo and M. Lapata, "Unsupervised opinion summarization with noising and denoising," 01 2020, pp. 1934–1945.

[51] R. K. Amplayo, S. Angelidis, and M. Lapata, "Aspect-controllable opinion summarization," *arXiv preprint arXiv:2109.03171*, 2021.

[52] A. Bražinskas, M. Lapata, and I. Titov, "Unsupervised opinion summarization as copycat-review generation," *arXiv preprint arXiv:1911.02247*, 2019.

[53] ——, "Few-shot learning for opinion summarization," *arXiv preprint arXiv:2004.14884*, 2020.

[54] "Find survey participants today." [Online]. Available: https://surveyswap.io/students

APPENDIX A
EXAMPLE SUMMARIES

A.1 Amazon

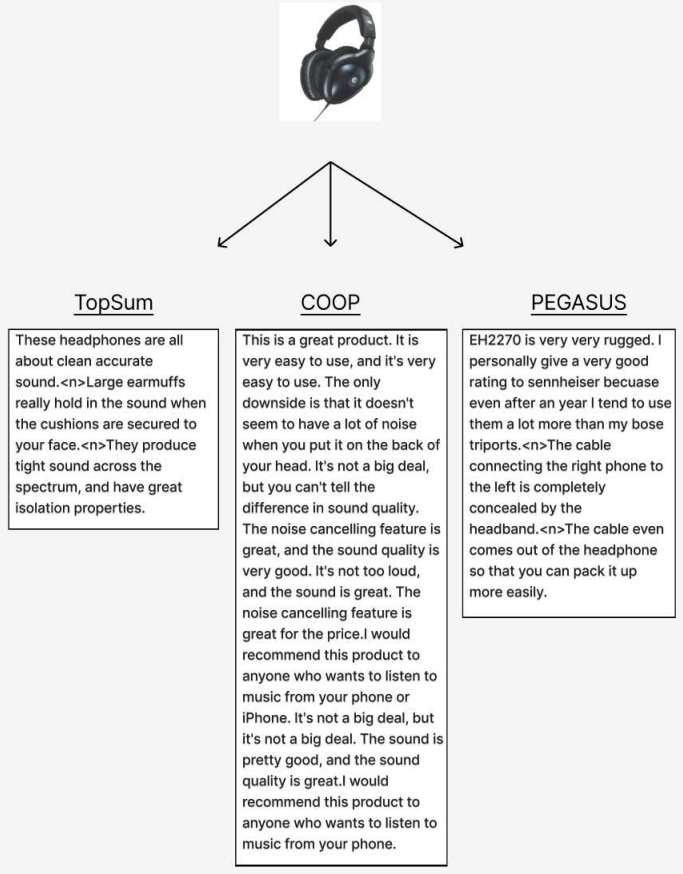

Fig. 11: Amazon Product B00005I9S3 Summaries

A.2 Yelp

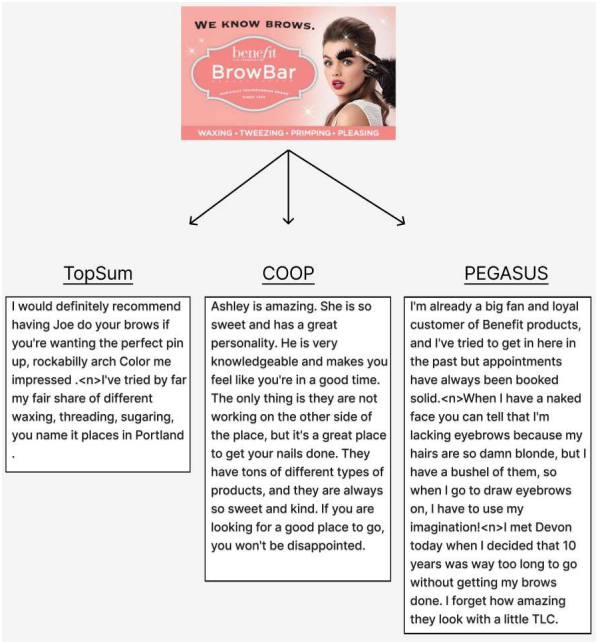

Fig. 12: Yelp Business 'Benefit Brow Bar at Ulta' Summaries

APPENDIX B
EXAMPLE SURVEY QUESTION

APC UPS Battery Replacement

Review 1: 'Like others who reviewed this item. I purchased this item for a client thinking I was getting a Genuine APC Replacement Cartridge but received a Generic Version of it instead. I was halfway expecting this anyway but with the description and picture indicating APC product so I kept my fingers crossed. Unit arrived in good condition although the shipping was a little slow. I was able to hot-swap the unit and only time will tell if the unit will hold up. Seems to be working ok for now.. I think there should be a indication on the product description that this is NOT a genuine APC Product.'

Review 2: 'good deeal'

Review 3: 'Shipped in a timely fashion, arrived in great shape and works. What more could I ask for? It was a great purchase experience, love it!'

Review 4: 'Easiest UPS fix I have ever experienced.'

Review 5: "As other reviews have said, this is not an APC manufactured product. Also, I had a power failure today and my UPS shut down immediately. When the power returned, the "Replace Battery" light was on. That means the battery lasted 2 1/2 years. That's less than the claimed 3-5 years. I've only had about 2 other power outages during that time so that's poor performance."

Review 6: "We have 11 of the APC 1500's that use the RBC-7 battery pack and when buying from Amazon we always get the same good and reliable service as if buying directly from APC at a more expensive cost. I highly recommend Amazon.Com for this purchase."

Review 7: "Fast delivery (two days to Bay Area) This rehabbed by APC 1500, which is now my son's"

Review 8: 'Vendor sent off-brand product instead of branded product shown - but did make full refund after I complained and made product return easy.'

Summary 1: "You have to use the battery that you would get at the same time. It's good for the price and the Amazon service from Amazon is very good as well.I highly recommend buying from this seller."

Summary 2: 'I purchased this item for a client thinking I was getting a Genuine APC Replacement Cartridge but received a Generic Version of it instead. I was halfway expecting this anyway but with the description and picture indicating APC product so I kept my fingers crossed. Unit arrived in good condition although the shipping was a little slow. I was able to hot-swap the unit and only time will tell if the unit will hold up. Seems to be working ok for now..'

Summary 3: "I highly recommend Amazon Easiest UPS fix I have ever experienced Fast delivery (two days to Bay Area) This rehabbed by APC 1500, which is now my son's I highly recommend Amazon Easiest UPS fix I have ever experienced Fast delivery (two days to Bay Area)"

Which order would you choose when ranking the summaries from most to least informative?

- Summary 1, Summary 2, Summary 3
- Summary 1, Summary 3, Summary 2
- Summary 2, Summary 1, Summary 3
- Summary 2, Summary 3, Summary 1
- Summary 3, Summary 1, Summary 2
- Summary 3, Summary 2, Summary 1