Generating Extractive Sentiment Summaries for Natural Language User Queries on Products

Siqi Gao
Computer Science Department
Brigham Young University
Provo, Utah 84602, USA
sophiegao99@gmail.com

Yiu-Kai Ng
Computer Science Department
Brigham Young University
Provo, Utah 84602, USA
ng@compsci.byu.edu

ABSTRACT

Multi-document sentiment analysis is an important natural language processing problem. Summaries generated by these analyzers can greatly reduce the time necessary to read a collection of topically-related documents to locate the desired information needs of a user. With the ever-increasing globalization and technology of the modern day, analysis of online user reviews on different products is an especially pertinent application of the aforementioned problem. At present there are way too many user reviews on popular products for potential buyers to spend adequate time to read and extract the most salient product details and opinions of previous buyers. In solving this problem, we propose a fully-automated summarizer to reduce the workload of online customers. The proposed system takes a user query and extracts the most relevant and essential comments made by individual reviewers. As opposed to existing multi-document summarization approaches, our summarizer compiles comprehensive reviews by extracting important facets and sentiment information based on various sentence features rather than applying complex machine learning algorithms. The design of our summarizer is easy to understand and implement, without the required massive training data and excessive training time. The conducted empirical study shows that the proposed summarization system outperforms current state-of-the-art multi-document sentiment summarization approaches.

CCS Concepts

•**Information systems** → **Information retrieval;** *Retrieval models and ranking; Similarity measures;* •**Applied computing** → **Document management and text processing;**

Keywords

Sentiment analysis; summarization; sentence features; facet identification, user reviews

1. INTRODUCTION

With the advance of the information technology, there is an abundance of data to be analyzed and understood, which is applicable to the task of processing sentiment of data. Extracting the opinions contained in the document text, however, is not a straightforward nor simple task. Sometimes the author of these documents can have mixed feelings about the topic of concern overall, or like some features of a product but not others. For this reason, multi-document sentiment analysis is a challenging and time-intensive process for people to handle manually. Research has shown that the average product review is 582 characters in length, with some outliers at upwards of 30,000 [41] and an average adult can read about 987 characters per minute [39]. It would take an ordinary person about 6 minutes to read 10 reviews. To appropriately form a justified opinion, a user would need to thoroughly investigate several related products in an attempt to select the best one. With the time constraint, users tend to read only a few reviews or skim through them. In fact, over 51% of online customers read fewer than 5 reviews before purchasing a product [7]. This is problematic, since a user can miss important details and overvalue others.

To reduce the workload of analyzing sentiment embedded in reviews and the amount of text to be considered, multi-document sentiment analysis systems have been developed in the past [4, 35]. These systems create a short snippet of the documents to be processed instead of the documents in their entirety, which offer the users a more reasonable time frame to extract the desired information. As the usefulness of such an approach is dependent upon the accuracy of representing the content of the snippets, it is critical for the summary of the reviews contain the most significant sentiments and salient details of the corresponding documents.

To facilitate the task of synthesizing opinions expressed in user reviews on a particular product, we introduce a query-based, multi-document sentiment summarization approach. Our solution is focused on generating multi-document sentiment summaries that are (i) cohesive, (ii) non-redundant, and (iii) diverse in terms of user opinion views. In addition, summaries generated by us achieve high coverage (of information included in the summary) and contain meaningful and relevant sentences. Our approach is simple and its summaries serve as a product guide to the users.

During the process of creating a summary, we (i) identify products, facets, and sentiment keywords in a query to determine the information needs of a user, (ii) detect different facets of a product P and cluster sentences in online reviews *on-the-fly*, (iii) find the *most-informative* sentences in

a top-rated review that capture the expressed opinions on P, and (iv) generate a *concise* summary of multiple reviews for answering the information need expressed in a user (sentiment) query. Our research work advances the technologies in developing summarization approaches on user reviews.

2. RELATED WORK

The area of study in extractive, sentimentally-representative summarization of online content has been conducted in the past. Huang et al. [17] present a summarization approach whereby "meaningful content" can be extracted from any arbitrary XML document for processing, and Bahrainian and Dengel [4] utilize extensive pre-processing on Twitter and other social media posts, which include removing URLs, stripping punctuation in hashtags, and splitting each tweet into smaller text segments based upon punctuation.

Ni et al. [31] devise a system using a trained classifier to select justification sentences from a review. These sentences contain facets of a product that assist a user in making a choice of favorability and other users in deciding between several potential options. Their system employed *BERT* developed by Devlin et al. [10] to fine-tune the label classifier. Aker et al. [2], on the other hand, propose a graph-based approach to labeling of topic clusters for any type of documents from reader comments to online news articles. They adopt Word2Vec word embeddings for clustering topic labels.

Jeong et al. [19] design a three-part system to extract keywords, generate an extractive summary of the text, and provide a simple search engine to help users find desired documents. Keywords have their importance estimated with statistical relevance weighting and are sorted thereupon. Ganesan and Zhai [13] introduce a special type of document ranking based on the individual qualities of multiple facets. The ranking, however, requires a specific type of query that consists of explicit sub-queries delimited by commas.

In generating a snippet for a document, the procedure outlined by He et al. [16] considers not only sentences that contain query keywords, but also selecting sentences for the snippet that are representative of the document as a whole. In analyzing sentiment, Farooq et al. [12] focus on the effect of negation on the sentiment polarity of a document. They identified and treated three types of negation differently: syntactic, morphological, and diminishers. Nallapati et al. [29] employ a summarization approach that relies on neural networks and is uniquely interpretable by allowing for the visualization of abstractive details. These document details include information content, salience, and novelty.

Intuitively, extractive summarization strategies are set up as binary classification problems with the goal to identify sentences in documents to be included in a summary. Lamsiyah et al. [22] use pre-trained sentence embedding methods and the centroid clustering method to provide analysis of sentences by clustering them using both semantic and syntactic relationships. Once clustered, sentences are weighted and selected based on their content relevance, uniqueness, and positioning. Bidoki et al. [5], on the other hand, combine statistical methods, machine learning, and graphical methods into a hybrid solution. Using the word2vec technique on this hybrid approach, the proposed summarizer is able to understand on the semantics of sentences in a document, and sentences are graphically clustered so that the prime summary sentences can be extracted from these clusters.

Mutlu et al. [28] define the boundaries of both "summary-worthy" and "summary-unworthy" sentences by parsing sentences into a vector based on different "features" of a document. The authors apply two different fussy systems to extract the summary-worthy sentences according to the feature vector. Instead of using document features, Tohalino et al. [38] employ a "multilayer network" approach that separates document into different layers such that each layer includes "nodes" representing sentences and the nodes being linked together by "edges" representing shared words or phrases. By differentiating between intra-layer edges and inter-layer edges, the accuracy of the resultant summary can be improved. Moreover, in creating a solution for the multi-document summarization problem, the authors of [34] develop the Multi-Objective Artificial Bee Colony algorithm. The algorithm imitates the behavior of honeybee swarms, focusing on creating summaries that cover as much of the document's content as possible while minimizing the number of redundant statements.

All of the existing extractive summarization approaches mentioned in this section, however, are significantly different from our proposed summarizer, since the former do not extract important facets and sentiment information presented in documents, which are essential in clustering relevant and sentiment-based comments on products.

3. OUR SENTIMENT SUMMARIZER

Our sentiment summarizer on user reviewers addresses various key design issues of a summarization system, which include simplicity to build, easy to use, and capable of capturing essential information. Our summarizer does not rely on complex machine learning algorithms. Although very helpful in different applications, machine learning approaches can be complicated to design and develop and require a training process using abundance of training data before becoming functional. We introduce a simple, and yet effective, sentiment summarization system that tailors towards the information needs of users. The information filtering and sentiment analysis procedures are straightforward, which are simply based on sentence scoring and ranking. We apply part-of-speech tagging for facet detection, various sentence heuristics to compute the score of a sentence in user reviews to determine its degree of significance in capturing essential information, and a sentence clustering approach to *avoid redundancy* and *maximize* the *coverage* of information included in a summary that meet the user's information needs, which are the novelty of our summarizer.

3.1 Identifying Users' Information Needs

Given a user query Q, which inquires on feedback on a particular product[1], we detect and segregate non-stop (key)words in Q, a process which identifies *products* and

[1]Since the design goal of our sentiment summarization approach is to synthesize archived feedback provided by web users on services and products, we process only queries with products explicitly specified.

facets that describe different aspects of a product, and filters *sentiment keywords* from *non-essential* ones such as stopwords, to recognize the information needs specified in Q. In accomplishing this task, we adopt a one-against-all [25] implementation of a multi-class SVM to identify information needs expressed in a query, which is a robust methodology that achieves state-of-the-art performance on classification.

To develop a multi-class SVM, each instance of the SVM is an input vector of a non-numerical, non-stopword[2] K in a query Q and is a succession of '1' ('0', respectively), each of which represents the presence (absence, respectively) of an SVM feature F (defined by us below) if F applies (does not apply, respectively) to K.

- **Capitalized** is set if the first letter of K is capitalized. The first character of a *product* is often capitalized.
- **Adjective** is set if K is given an *adjective* part-of-speech (POS) tag. We employ the Stanford POS tagger[3] for assigning POS tags, such as noun, verb, adjective, etc., to keywords (in Q). *Sentiment keywords* specified in Q are often adjectives which describe different aspects of a product specified in Q.
- **Sentiment** is set if K is a *sentiment keyword*, which is determined by using a list of more than 4,000 sentiment keywords provided by the General Inquirer[4].
- **After-Preposition** is set if K appears immediately after a preposition, identified using the *Stanford POS* tagger. Both *products* and *facets* tend to occur after a preposition in Q.
- **After-Apostrophe** is set if K appears immediately after a term in a Saxon genitive form, i.e., a traditional term for the apostrophe-s. *Facets* often appear after a term in the Saxon genitive form in Q.
- **Before-Sentiment** is set if K appears immediately before a sentiment keyword in Q. Both *products* and *facets* are often followed by a sentiment keyword in Q.
- **Stopword** is set if K is a stopword, which is a *non-essential* term. We compiled our own list of 865 stopwords using multiple stopword lists posted online for this feature.
- **Is-5W1H** is set if K is one of the keywords frequently used in formulating questions, i.e., "what", "when", "where", "who", "why", and "how". 5W1H terms are treated as *non-essential* terms, since "when", "where", "who", and "why" do not appear often in sentiment questions, whereas "what" and "how", which appear more often, do not have a direct impact on the information needs specified in users' questions.

To verify that each of the chosen SVM features listed above is accurate in identifying keywords (in users' queries) that are either *Products*, *Facets*, *Sentiment Keywords*, or *Non-Essential Terms*, we conducted an empirical study using a dataset, denoted *Property-DS*, which does not overlap with the dataset introduced in Section 4.1, for analyzing the performance of our multi-class SVM. *Property-DS* consists of 3,000 opinion questions randomly extracted from Yahoo! Answers[5] and WikiAnswers[6]. Keywords in each of the questions in *Property-DS* were identified as products, facets, sentiment keywords, or non-essential terms by independent assessors prior to conducting the evaluation.

Table 1 shows the types of SVM features that are supposed to be identified, and we computed the percentages of keywords (in the questions in *Property-DS*) belonged to each keyword type, i.e., *Products, Facets, Sentiment Keywords*, and *Non-Essential Terms*, that are accurately identified by the aforementioned SVM features. The accuracy ratios of identifying *Product*-type keywords in *Property-DS* are 96%, 92%, and 83%, respectively that are either *capitalized*, appear *after* a *preposition*, or appear *before* a *sentiment* keyword, whereas the percentage of misclassified *products* in the 3,000 questions in *Property-DS* identified by each remaining SVM feature is below 15%. For the *Facet* keywords, the accuracy ratios are 80%, 90%, and 85% for appearing *after* a *preposition*, *after* an apostrophe, and *before* a *sentiment* keyword, respectively, whereas the accuracy ratio for detecting *Sentiment*-type keywords are 90% and 97% for appearing as *adjective* and *sentiment*, respectively. As expected, our SVM achieves a 100% accuracy in recognizing all the *stopwords* and *5WIH* keywords as non-essential terms. The empirical study validates that the chosen SVM features used by our multi-class SVM adequately classify the types of keywords as designed.

3.2 Creating Sentence Clusters

In this section, we first introduce our approaches in creating and ranking sentence cluster labels for downstream processing. Hereafter, we discuss our strategy in choosing sentences extracted from user reviews that are assigned to different labeled clusters to be included in the summary generated in response to a user query.

3.2.1 Creating Cluster Labels

We create concise and accurate cluster labels that reflect the *facets* mentioned in the top-100 user reviews[7], denoted *TopRev*, using the suffix array algorithm, which has been proved to be *efficient* and *effective* in discovering key phrases in large text collections [8]. The algorithm generates a list of cluster labels by simply extracting all the suffixes in reviews that are sorted alphabetically. Since the generated list of suffixes may include labels that are not representative of facets describing the product P in *TopRev*, we removes labels that (i) are *numeric*, (ii) cross sentence *boundaries*, since sentence markers indicate a topical shift, (iii) are *incomplete*, i.e., included as substrings in other labels, (iv) end in the *Saxon genitive form*, or (v) are *sentiment* keywords, i.e., terms that express a positive or negative polarity (which are

[2] *Stopwords* are commonly-occurred words, such as articles, prepositions, and conjunctions, which carry little meaning.
[3] nlp.stanford.edu/software/tagger.shtml
[4] wjh.harvard.edu/~inquirer/homecat.htm
[5] answers.yahoo.com
[6] www.answers.com
[7] One hundred reviews is an *ideal* set for creating summaries [11].

Table 1: Keyword types that are identified by each of the previously introduced SVM features

SVM Features	Product	Facet	Sentiment Keyword	Non-Essential Term
Capitalized	X			
Adjective			X	
Sentiment			X	
After-Preposition	X	X		
After-Apostrophe		X		
Before-Sentiment	X	X		
Stopword				X
5W1H				X

considered only in generating our sentiment summaries).

3.2.2 Ranking Cluster Labels

To capture the content-significance of the created cluster labels, we proceed to rank the labels using various measures, which are effective in identifying representative cluster labels and are defined as follows:

- The **frequency** of a label L, denoted $Freq(L)$, reflects the *frequency of occurrence* of L in the top-100 user reviews $TopRev$. The higher $Freq(L)$ is, the higher the ranking position of L among the cluster labels.

- The **stability** of a label, denoted $Stability(L)$, measures the *mutual information* (i.e., dependence[8]) of L. Given that L may contain multiple keywords, i.e., L = "$c_1 \ldots c_n$", where c_i $(1 \leq i \leq n)$ is a keyword in L, $Stability(L)$ is defined as

$$Stability(L) = \frac{f(L)}{f(L_L) + f(L_R) - f(L)} \quad (1)$$

  where L_L = "$c_1 \ldots c_{n-1}$", L_R = "$c_2 \ldots c_n$", and $f(L)$, $f(L_L)$, and $f(L_R)$ are the *frequencies of occurrence* of L, L_L, and L_R, respectively in $TopRev$.

- The **significance** of a label L, $Sig\_L(L)$, is a function that assigns *more* weight to *longer* cluster labels, since longer labels are more meaningful, i.e., descriptive.

$$Sig\_L(L) = f(L) \times g(|L|) \quad (2)$$

  where $f(L)$ is the *frequency of occurrence* of L in $TopRev$, $|L|$ is the number of keywords in L, and $g(x)$ is a function such that $g(1) = 0$, $g(x) = log_2\ x$ (if $2 \leq x \leq 8$), and $g(x) = 3$ (if $x > 8$).

We compute a *ranking score* for each cluster label L, denoted $LRank(L)$, which reflects the significance of L in capturing the content of the reviews in $TopRev$, by combining *Freq*, *Stability*, and Sig_L[9] of L using the *Stanford Certainty Factor* [26].

$$LRank(L) = \frac{Freq(L) + Stability(L) + Sig\_L(L)}{1 - Min\{Freq(L), Stability(L), Sig\_L(L)\}} \quad (3)$$

3.2.3 Assigning Sentences to Clusters

Using the set of identified cluster labels, we assign sentences in *TopRev* to different clusters using the word-correlation factors. The *word-correlation factors* (wcf) in our *word-similarity* matrix, denoted $WS\text{-}matrix$, is a 54,625 × 54,625 symmetric matrix. $WS\text{-}matrix$ was generated using a set of approximately 880,000 Wikipedia documents[10] written by more than 89,000 authors on various topics and writing styles. The wcf of any two words[11] is computed using their (i) *frequency of co-occurrence* and (ii) *relative distances* in each Wikipedia document.

$$wcf(i,j) = \frac{\sum_{D \in Wiki} \left(\frac{\sum_{k_i \in D} \sum_{k_j \in D} \frac{1}{d(k_i,k_j)+1}}{N_i \times N_j} \right)}{|Wiki|} \quad (4)$$

where $|Wiki|$ is the number of documents in the Wikipedia collection, i.e., $Wiki$, $d(k_i, k_j)$ denotes the *distance* (i.e., the number of words in) between words i and j or their stems in a Wiki document D in which they co-occur, and N_i (N_j, respectively) is the number of times word i (j, respectively) and its *stems* variations appeared in D. Compared with WordNet[12] in which each pair of words is not assigned a *similarity weight*, word-correlation factors offer a more sophisticated measure of word similarity.

To cluster sentences in *TopRev* that address the same or similar facets, we compute the *degree of similarity* between each sentence S and label L in the set of identified cluster labels as follows:

[8] *Dependence* identifies labels that characterize the contents of sentences in one cluster in contrast to others. The higher the *mutual* information of L is, the more *dependent* L is as a cluster label.

[9] Since *Freq*, *Stability*, and Sig_L are in different numerical scales, we first normalize the values using a logarithmic equation so that they are in the same range.

[10] en.wikipedia.org/wiki/Wikipedia:Database_download

[11] Words in the Wikipedia documents were *stemmed*, i.e., reduced to their grammatical roots, after the stopwords were removed which, as an effect, minimize the number of keywords to be considered.

[12] wordnet.princeton.edu

$$LS\_Sim(L,S) = \frac{\sum_{i=1}^{|S|}\sum_{j=1}^{|L|} wcf(w_i,w_j)}{|S|} \quad (5)$$

where $|S|$ ($|L|$, respectively) is the number of words in S (L, respectively), w_i (w_j, respectively) is a keyword in S (L, respectively), and $wcf(w_i, w_j)$ is the word-correlation factor of w_i and w_j. Since the longer S is, the higher $LS\_Sim(L, S)$ is, we normalize $LS\_Sim(L, S)$ by dividing the accumulated word-correlation factors by the number of words in S, i.e., $|S|$.

Having computed $LS\_Sim(L, S)$, S is assigned to the cluster C with label L_C such that the $LS\_Sim(L_C, S)$ score is the *highest* among all the LS_Sim scores of S and other labels.

3.3 Ranking Sentences in Clusters

Each sentence S in *TopRev*, is assigned a *relevance score*, denoted RS, which indicates its relative significance in capturing the content of the reviews in *TopRev*. To compute RS of S, we utilize the sentence *features* presented between Section 3.3.1 and Section 3.3.6.

3.3.1 Significance Factor

We rank each sentence S in a *TopRev* review using *significance factor* [8]. The *significance factor* for S relays how significant S is based on the significance of the words in S. *Significant words* are defined as words of medium frequency in the reviews, where *medium* means that the frequency is between predefined high-frequency and low-frequency cutoff values. Intuitively, higher scores are given to sentences with more *significant words*. Given that $f_{r,w}$ is the *frequency* of word w in the review r, then w is a significant word if (i) it is not a stopword, which eliminates the high-frequency, non-essential words, and (ii)

$$f_{r,w} \geq \begin{cases} 7 - 0.1 \times (25 - Z) & \text{if } Z < 25 \\ 7 & \text{if } 25 \leq Z \leq 40 \\ 7 + 0.1 \times (Z - 40) & \text{otherwise} \end{cases} \quad (6)$$

where Z is the number of sentences in r, and 25 and 40 are the low- and high-frequency cutoff values, respectively.

Once we know which words in a user review are significant, we can calculate the significance factor (SF) of a sentence S, i.e.,

$$SF(S) = \frac{|significant\text{-}words|^2}{|S|}, \quad (7)$$

where $|S|$ is the number of words in S and $|significant\text{-}words|$ is the number of significant words in S.

3.3.2 Sentiment Value

The *sentiment value* of a sentence S is determined by using SentiWordNet, a lexical resource in which a word w is associated with three numerical scores, i.e., $Obj(w)$, $Pos(w)$, and $Neg(w)$, describing how Objective (i.e., neutral), Positive, and Negative w are. We compute the *sentiment value* of S by computing the absolute value of the *sum* of $Obj(w)$, $Pos(w)$, and $Neg(w)$ of each non-stopword w in S.

3.3.3 Sentence-Label Similarity

Since a cluster label L captures the facet of a product specified in the sentences of its cluster C, we compute the SLB_Sim score that measures the *degree of resemblance* between the label L (of C) and each sentence S in C as

$$SLB\_Sim(L,S) = \frac{\sum_{i=1}^{|S|}\sum_{j=1}^{|L|} wcf(w_i,w_j)}{|S|} \quad (8)$$

where $|S|$ ($|L|$, respectively) is the number of words in S (L, respectively), w_i (w_j, respectively) is a keyword in S (L, respectively), and $wcf(w_i, w_j)$ is the word-correlation factor of w_i and w_j. As the length of S can potentially affect $SLB\_Sim(L, S)$, since the longer S is, the higher $SLB\_Sim(L, S)$ is, the accumulated word-correlation factors of $SLB\_Sim(L, S)$ is divided by the number of words in S.

3.3.4 Sentence-to-Sentence Similarity

In order to avoid choosing (very) similar sentences to be included in a summary, we prioritize sentences that are unique based on the wcf value of the words in each sentence in *TopRev*. The *degree of similarity* of a sentence S_i with respect to the others in *TopRev* indicates the relative degree of S_i in capturing the overall semantic *content* of *TopRev*, denoted $Sim(S_i)$. We compute $Sim(S_i)$ using (i) the wcf of every word in S_i and words in each remaining sentence S_j in *TopRev* and (ii) the *Odds ratio* $= \frac{p}{1-p}$ [20].

$$Sim(S_i) = \frac{\sum_{j=1,i\neq j}^{|S|}\sum_{k=1}^{n}\sum_{l=1}^{m} wcf(w_k,w_l)}{1 - \sum_{j=1,i\neq j}^{|S|}\sum_{k=1}^{n}\sum_{l=1}^{m} wcf(w_k,\ w_l)} \quad (9)$$

where $|S|$ is the number of sentences in *TopRev*, n (m, respectively) is the number of words in S_i (S_j, respectively), which is a sentence in *TopRev*, and w_k (w_l, respectively) is a word in S_i (S_j, respectively).

3.3.5 Sentence Length

We penalize sentences that are either *too short* (< 15 words) or *too long* (> 30 words) [35]. Short sentences are detrimental to our summarization task, since they require some introduction or do not have as much information included, whereas long sentences have a higher probability of discussing multiple topics and can be found somewhere else in a user review. We compute the *Sentence Length*, denoted SL, of a sentence S as

$$SL(S) = \begin{cases} -1 & \text{if } |S| < 15 \text{ or } |S| > 30 \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where $|S|$ is the number of (stop)words in S.

3.3.6 Named Entity

An entity can be any word or a series of words that consistently references to the same concept. It is well-known that a sentence that contains a named entity usually captures useful information in a document [32]. Named Entity Recognition (NER) focuses on (i) determining if a word w is part of a named entity and (ii) assigning w to the correct

entity. In Natural Language Processing (NLP), this can be accomplished by categorizing each word w into a category, such as a person, organization, time, and location, assuming that w belongs to a name entity. To determine the *named entity weight* of a sentence S, denoted $NE(S)$, we consider the number of named entities in S. By summing the number of named entities in S, we can prioritize sentences that are more informative, i.e., with more named entities, than others.

$$NE(S) = \frac{\sum_{i=1}^{|E|} f(E_i)}{f(E)} \tag{11}$$

where $|E|$ is the number of named entities in S, $f(E_i)$ is the frequency of occurrence of entity E_i in *TopRev* T, and $f(E)$ is the sum of the frequency of occurrence of all named entities in T.

3.3.7 CombMNZ

Based on the respective scores of the features discussed in Sections 3.3.1 through 3.3.6 that are computed for each sentence in a user review, we rank all the sentences in the top-100 user reviews accordingly. To compute a single score on which the cumulative effect of the six different features of each sentence are used for ranking propose, we rely on the CombMNZ model. CombMNZ is a well-established data fusion method for combining multiple ranked lists on an item I, i.e., a sentence in our case, to determine a *joint* ranking of I, a well-known rank-aggregation task or data fusion task.

$$CombMNZ_I = \sum_{c=1}^{N} I^c \times |I^c > 0|, \text{ where } I^c = \frac{S^I - I^c_{min}}{I^c_{max} - I^c_{min}} \tag{12}$$

where N is the number of ranked lists to be fused, which is *six* in our case, I^c is the normalized score of I in the ranked list c, and $|I^c > 0|$ is the number of non-zero, normalized scores of I in the lists to be fused. Prior to computing the ranking score of a sentence S, we transform the original scores in each feature ranked list of S into a *common range* [0, 1] such that S^I is the score of I in the ranked list c to be normalized, I^c_{max} (I^c_{min}, respectively) is the maximum (minimum, respectively) score available in c.

3.4 Our Approach in Creating Summaries

In creating a user review summary, we include some sentences in clusters created in Section 3.2.3. To determine which sentences are to be extracted from which cluster and included in the summary, we rely on the ranked cluster labels introduced in Section 3.2.2. Using the ranked labels, we include in the summary *Sum* of a user query one sentence from a cluster at a time, starting from the cluster with the *highest-ranked label* (based on its *LRank* score defined in Equation 3), up till the limit of the summary size is reached. Note that the selection terminates whenever the length of the sentences that are already included in *Sum* is less than 250 words, and the addition of the newly-selected sentence causes the length to exceeds *250* words, which is recommended by the *Text Analysis Conference (TAC)*[13] for a multi-document summary.

[13]nist.gov/tac

If the number of sentences that should be included in a summary *exceeds* the number of generated clusters after selecting the highest-ranked sentence in each cluster, then in the subsequent iterations we select the next highest-ranked sentence S' in cluster C with the *lowest similarity score*, denoted LSS, with respect to the sentence(s) S in C that has (have) already been included in the summary for Q. The LSS score of S' is computed as the *sum* of the word-correlation factors between each non-stop, stemmed word in S' and S. By considering the LSS score of a sentence S' in a cluster with respect to S in the summary being constructed for Q, we ensure that S and S' are *distinct* in contents, which *avoids redundancy* and *maximizes coverage* in terms of information included in the summary, a novelty of our summarization approach. Moreover, if a facet F is preceded by a negation term in Q, any cluster label that includes F is excluded from the sentence selection process.

3.5 Generating Different Types of Summaries

We create a single summary in response to the information needs specified in a user query Q. A summary is (i) *General*, if Q inquires on common feedback of a particular product P, (ii) *Sentiment-Specific*, if Q asks for positive or negative information about P, (iii) *Facet-Specific*, if Q queries on specific facets of P, or (iv) *Facet-Sentiment-Specific*, if Q looks for sentiment information on specific facets of P.

3.5.1 General Summaries

A *General* summary addresses different facets and sentiments of a product being reviewed. It consists of the highest-ranked sentences (regardless of their polarity), along with probably the ones with the lowest LSS scores, in (highly ranked) clusters (with the highly ranked labels determined using Equation 3) created in Section 3.2.3, which are chosen by following the procedure established in Section 3.4 to be included in the summary.

3.5.2 Sentiment-Specific Summaries

A *Sentiment-Specific* summary is created in the same manner as a *General* summary, except that only the highly-ranked sentences (along with probably the ones with the lowest LSS score) in clusters which satisfy the *sentiment* (i.e., positive or negative) specified in Q (identified using the keyword tagger introduced in Section 3.1) are included in the summary for Q. To determine the positive or negative polarity of a sentence S in a cluster, we calculate the overall sentiment score of S by *subtracting* the sum of its *negative* word SentiWordNet scores from the sum of its *positive* word SentiWordNet scores that reflects the (degree of) sentiment of S such that if its sentiment score is positive (negative, respectively), then S is labeled as positive (negative, respectively). Employing this sentence-based, sentiment approach, we include in each *Sentiment-Specific* (*Facet-Sentiment-Specific*, respectively as introduced in Section 3.5.4) summary sentences reflecting the sentiment specified in the corresponding query.

3.5.3 Facet-Specific Summaries

To create the *Facet-Specific* summary for Q, we first identify

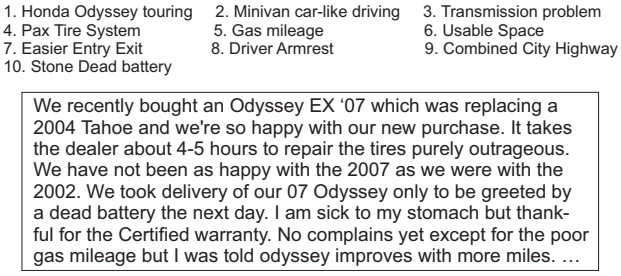


Figure 1: The 10 different cluster labels and a portion of the *Facet-Sentiment-Specific* summary created for the query "The pros and cons of 2007 Honda Odyssey tire and battery"

the labels (from the set of labels created in Section 3.2.1) that are highly similar to each of the facets F (determined using the keyword tagger in Section 3.1) specified in Q. To identify cluster labels *highly similar* to F, we employ a reduced version of the word-similarity matrix which contains 13% of the most frequently-occurring words (based on their *frequencies of occurrence* in the Wikipedia documents), and for the remaining 87% of the less-frequently-occurring words, only exact-matched correlation factors, i.e., word correlation factors of values 1.0, are used. A label L (and its cluster) is considered highly similar to F only if the word-correlation factor of (w_1, w_2) between the non-stop, stemmed word w_1 in L and w_2 in F exists in the *reduced* matrix, which significantly minimizes the processing time to identify the desired labels without affecting the quality of the created summaries.

In creating the *Facet-Specific* summary, we follow the same procedure as detailed in Section 3.4 for selecting sentences, regardless of their polarity, to be included in the summary. Instead of considering ranked labels, we rely on the *ranking* of the *highly similar* labels with respect to F computed using the reduced word-correlation matrix. Moreover, the content of the *Facet-Specific* summary of Q is uniformly divided among each of the facets specified in Q, i.e., the number of sentences to be included in the summary is uniformly extracted from the sentence clusters with labels highly-similar to or the same as each facet specified in Q.

3.5.4 Facet-Sentiment-Specific Summaries

The *Facet-Sentiment-Specific* summary is created in response to Q by including solely the sentences in clusters which (i) reflect the polarity specified in Q (as detailed in Section 3.5.2) and (ii) belong to the clusters with labels highly similar to or the same as a facet in Q (as in a *Facet-Specific* summary). The process of including sentences in a *Facet-Sentiment-Specific* summary is the same as the one detailed in section 3.5.3 for *Facet-Specific* summaries, except that only sentences with the same polarity as the one specified in Q are included in the summary. Figure 1 depicts the ten cluster labels and the *Facet-Sentiment-Specific* summary created by using our summarizer for the query, "The pros and cons of 2007 Honda Odyssey tire and battery."

4. EXPERIMENTAL RESULTS

To assess the performance of our multi-document sentiment summarization approach, we first present the datasets used for the empirical study (in Section 4.1) and define the evaluation metrics used for analyzing the performance of our summarizer (in Section 4.2). Hereafter, we compared summaries created by our summarizer and a number of well-known extractive summarization approaches (in Section 4.2.2) and introduce the *criteria* for evaluating the *quality* of generated summaries using different measures (in Sections 4.3.2 and 4.3.5) for comparison purpose.

4.1 Datasets

We rely on the benchmark dataset set up for the Opinion Summarization Pilot task of the 2008 Text Analysis Conference, denoted TAC-08[14], a dataset that includes a set of 87 (squishy-list) questions, to assess the effectiveness of our sentiment summarization approach in (i) identifying information needs specified in a user's query Q and (ii) creating summaries that satisfy the information specified in Q. For each question Q in TAC-08, which is treated as a query, there is (i) a set of documents D (extracted from the TREC Blog06 collection[15]) that serves as the source for creating a summary (of D) and (ii) a list of *expert-created* sentences/phrases that are expected to be included in a summary (of D) with respect to Q. Since the goal of the Opinion Summarization task is to evaluate multi-document sentiment summaries created in response to a question, TAC-08 is ideal for evaluating the effectiveness of our sentiment summarization approach, including keyword tagging.

4.2 Performance Evaluation

We define in this section the various evaluation metrics used for assessing our summarization approach in (i) identifying the information needs expressed in a query Q (in Section 4.2.1) and (ii) generating a summary that satisfies Q (in Section 4.2.2)

4.2.1 Accuracy on Keyword Tagging

To assess the performance of the multi-class SVM adopted by our sentiment summarization approach for *identifying* the types of keywords expressed in users' queries, as well as keywords in the user reviews, we compute the *accuracy* ratio, which is defined as the proportion of the number of keywords *correctly identified* by the multi-class SVM as *products, facets, sentiment keywords, or non-essential terms* over the total number of keywords used for evaluating the SVM, i.e., the number of keywords in the 87 queries in TAC-08 and the TREC Blog06 collection, which are treated as user reviews for our empirical study.

To create the multi-class SVM (introduced in Section 3.1) for identifying products, facets, sentiment keywords, and non-essential terms in users' queries and reviews, we constructed a dataset, denoted *SVM-Data*, which consists of approximately 32,000 keywords extracted from 300 (opinion) queries and 2,800 responses, in addition to the TAC-08 and TREC Bolg06, which were employed as user queries and reviews, retrieved from WikiAnswers and Yahoo! Answers. To vali-

[14] www.nist.gov/tac/data/index.html

[15] TREC Blog06 is a collection of blog posts downloaded from the Web between December 2005 and February 2006.

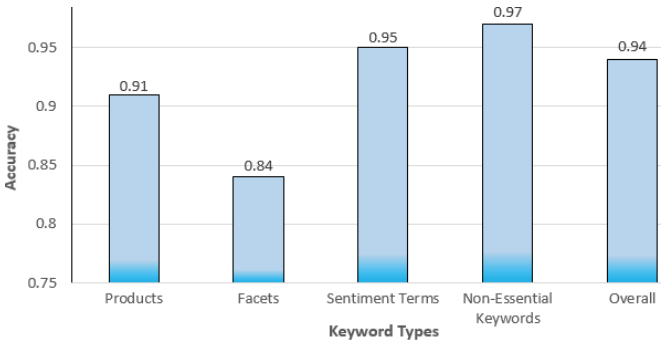


Figure 2: (Overall) Accuracy of our multi-class SVM for identifying users' information needs specified in TAC-08 query set and keywords in Blog06 collection and SVM-Data

date the effectiveness of the multi-class SVM[16], we adopted the 10-fold cross-validation approach so that in each of the 10 repetitions, 90% of the instances in *SVM-Data* were used for classification and the remaining 10% for validation purpose.

The *accuracy* of our sentiment summarization approach in identifying the types of keywords specified in user queries and keywords in reviews was computed based on the trained multi-class SVM on each keyword in each of the 87 queries in TAC-08 and the user reviews in the TREC Blog06 collection. As shown in Figure 2, the overall accuracy of the multi-class SVM on identifying the keywords in TAC-08 queries, Blog06 posts and SVM-Data is 94%. The accuracy ratios achieved on correctly detecting products, facets, sentiment keywords, and non-essential terms, respectively, are also shown in Figure 2. Note that the low accuracy ratio of facets as shown in the figure, in comparing with products, sentiment keywords, and non-essential terms, is due to the fact that facets are more difficult to classify than the rest because of their variations; however, the accuracy ratio for identifying facets is still in the mid-80% range, which is a high percentage.

4.2.2 Nugget Pyramid Score

To verify whether our generated summaries *satisfy the information needs specified in user queries*, we rely on the *Nugget-Pyramid* score [24]. Consider a test query Q in TAC-08, "How good is a town house in Brooklyn?" A *human assessor* creates a list of relevant *nuggets*, which are expert-created phrases/sentences, e.g., "Brooklyn is livable," and "As a current owner of a town house in Brooklyn, I feel good about its safety", that address different aspects of Q. It is expected that a "good" summary includes (the majority of the) nuggets in the corresponding list of *relevant nuggets*. We evaluate a summary S generated by our approach in response to Q by verifying the (non-)existence of a *conceptual match* between each provided nugget and S, which is a match independent of the distinct wording used in S and the nugget.

The *Nugget-Pyramid* score of S [24], which is the *weighted harmonic mean* between (*nugget*) *precision* and (*nugget*) *recall* that favors recall (which is controlled by a parameter β that is set to 3 based on our empirical study), is calculated as

$$Nugget\_Pyramid(S) = \frac{(\beta^2+1) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \quad (13)$$

$$\text{Precision} = \begin{cases} 1 & \text{if } Length < Allowance \\ 1 - \frac{Length - Allowance}{Length} & \text{otherwise} \end{cases} \quad (14)$$

where $Allowance = 100 \times$ *the_number_of_nuggets_included_in_S* and *Length* is the total number of non-white-space characters in S.

$$Recall = \frac{\sum_{m \in A} w_m}{\sum_{n \in V} w_n} \quad (15)$$

where A is the set of (relevant) reference nuggets that are included in S, V is the set of all reference nuggets (as determined in TAC-08), and w_m (w_n, respectively) is the score (between 0 and 1, inclusively) of nugget m (n, respectively), which is determined by (*human*) *assessors*.

We have validated our approach in creating sentiment summaries that satisfy the information needs expressed in user queries using the Nugget-Pyramid score. Figure 3 shows the *minimum*, *maximum*, *median*, and *average* Nugget-Pyramid scores of our summarization approach based on TAC-08. In comparing the (average) Nugget-Pyramid scores achieved by 19[17] query-based multi-document summarizers of TAC-08, our summarization approach ranks *fourth*. Note that 36 multi-document summarizers originally took part at the 2008 TAC. However, 17 of the summarizers relied on *snippets*[18] of information provided by TAC in creating the summaries. To perform unbiased comparisons, we only consider the 19 summarizers that operate in a manner similar to our approach, i.e., they do not rely on external information, such as snippets, in creating a summary. In addition, we have evaluated 22 summaries, as opposed to the 87 summaries created using the questions and documents in TAC-08. The choice follows the evaluation premises defined by TAC, which provides assessment, in terms of using the computed Nugget-Pyramid scores, for only 22 summaries generated by each of the 19 automatic multi-document summarizers. The three summarizers that are ranked higher than our summarizer have achieved a Nugget-Pyramid score of 0.251, 0.223, and 0.201, respectively, which are close to the average score, 0.190, of our summarizer. This results have verified the effectiveness of our summarizer in generating summaries that satisfy users' information needs.

4.3 Comparing the Performance of Various Extractive Summarizers

To assess the effectiveness of our summarizer in generating high-quality multi-document summaries with respect to

[16]Keywords in *SVM-Data* were previously labeled by independent assessors as *Products*, *Facets*, *Sentiment Keywords*, or *Non-Essential Terms*.

[17]The Nugget-Pyramid score for the 19 summarizers are available at The TAC-08 site www.nist.gov/tac/protected/past-blog06/2008/QA2008_runs.tar.gz.

[18]Snippets are answers to queries in TAC-08 generated by existing question-answering systems.

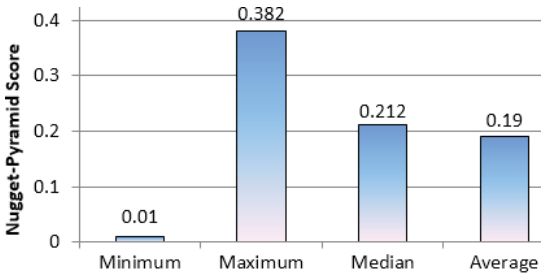


Figure 3: The *Nugget-Pyramid* scores achieved by our summarizer using TAC-08

user queries, we compare its performance with other existing baseline extractive summarization approaches (presented in Section 4.3.1) in terms of different *ROUGE* scores in Sections 4.3.3 and 4.3.4 and other linguistic quality measures in Section 4.3.5.

4.3.1 Baseline Extractive Summarization Methods

We have chosen four well-known extractive summarization systems: (i) Latent Semantic Analysis (LSA) [37], (ii) KL-Sum [15], (iii) SubBasic [30], and (iv) Luhn [36], as baseline models for performance evaluation against our extractive summarizer. These baseline models are presented below.

Latent Semantic Analysis (LSA). In choosing highly-ranked sentences for creating a multi-document summary in the news domain, Gong and Liu [14] propose an extractive summarization method using the latent semantic analysis (LSA), which was further enhanced by Steinberger et al. [37]. Using the LSA method, a *term-sentence matrix* (an $n \times m$ matrix) is first built, where each row corresponds to a *word* from the input document (with n distinct words) and each column corresponds to a *sentence* (m different sentences). In the matrix, each entry $a_{i,j}$ is the *weight* of the word i in sentence j, which is computed by using the TF-IDF approach [1]. Hereafter, *singular value decomposition* (SVD) is used on the *term-sentence* matrix to generate the matrix $A = U\Sigma V^T$, where U $(n \times m)$ represents a *term-topic* matrix with weights of words, Σ is a *diagonal* matrix $(m \times m)$ in which each row i denotes the *weight* of a topic i, V^T is the *topic-sentence* matrix, and $M = \Sigma V^T$ is the matrix that captures how much a sentence represents a topic such that $m_{i,j}$ denotes the *weight* of the topic i in sentence j.

Initially, the LSA proposed by [14] chooses one sentence per topic, and according to the length of a summary in terms of sentences, they retained the number of topics. However, a topic likely needs more than one sentence to convey its information. This approach was further improved by the enhancement introduced in [37] based on the "weight" of each sentence. The authors of [37] choose the sentences with greatest combined weight across all topics, possibly including more than one sentence about an important topic, rather than always choosing one sentence for each topic as done by Gong and Liu [14].

KL-Sum. As mentioned in [3], one of the limitations of the existing multi-document summarization systems is that sentence scores computed by them typically do not have very clear probabilistic interpretations, and heuristics are used instead in calculating many of the sentence scores. To enhance the topic/sentence and document representation using probability interpretation, topic/sentence models often utilize a distinct measure for scoring the sentence called Kullback-Leibler (KL). *KL-divergence* is a well-known method for scoring sentences in generating summaries, since it has proved the fact that *good* summaries are intuitively similar to the *input* documents based on the probability of word occurrence. The divergence indicates how the importance of words alters in the summary in comparison with the input, i.e., the KL-divergence of a *good* summary and the input will be low. By adopting the KL-divergence model, the KL-SUM algorithm [15] introduces a criterion for generating a summary S given a document collection D as follows:

$$S^* = \min_{S:words(S)\leq L} KL(P_D \parallel P_S) \tag{16}$$

where P_S is the candidate summary S, P_D is the set of reference summaries D, and $KL(P \parallel Q)$ denotes the Kullback-Leibler (KL) divergence given by

$$\sum_w P(w) log \frac{P(w)}{Q(w)} \tag{17}$$

where $P(w)$ and $Q(w)$ are probabilities of word w in P and Q, respectively. This quantity captures the divergence between the true distribution P and the approximating distribution Q (i.e., the summary distribution).

SumBasic. To generate a multi-document summary, SubBasic [30] relies solely on the frequency of words in the original text and is conceptually very simple. The design of SumBasic is motivated by the observation that words occurring *frequently* in a document cluster appear with higher probability in the human summaries than words occurring *less frequently*, the underlying premise of SumBasic. This premise explains why SumBasic computes the *weight* of a sentence S as equal to the *average probability* of the words in S.

SumBasic uses simple word probabilities with an update function to generate a summary for a number of documents [40]. To prevent including redundant sentences in the summary, SumBasic updates the probability of the *words* in the current sentence based on their occurrence in preceding selected sentences, a necessary mechanism since several documents might convey highly similar, or even identical, important sentences. This prevention approach allows the sentence alternatives to be considered independently, while maintaining redundancy at a minimum.

According to the performance analysis of various multi-document summarizers conducted by Inouye and Kalita [18], SumBasic, a simple frequency-based summarizer, performs better than summarizers that incorporated more information or more complexity both in F-measure scores and in human evaluation scores. Because the more complex algorithms did not perform as well, it seems that simple word frequency and redundancy reduction are the best techniques for summarizing various topics of different documents.

Luhn's Heuristic Summarization Approach. One of the first text summarization algorithms was proposed by Hans Luhn who introduced a systematic approach to perform summa-

rization which forms the core of the field today [27]. In this extraction-based summarization method, a sentence is assigned a *significance factor*[19] such that sentences with the highest significance factors are selected to be included in the summary of a collection of documents. The relative significance of each sentence in a document is captured with the number of significant words (its definition is given in Section 3.3.1) and their physical proximity within a sentence. Luhn proposed that the significance of each word in a document signifies how important it is. The idea is that any sentence with maximum occurrences of the highest frequency words, i.e., stopwords, and least occurrences are not important to the meaning of document than others.

Luhn's algorithm is a naive approach based on TF-IDF and looking at the "window size" of non-significant words between significant words of a document. It also assigns higher weights to sentences occurring near the beginning of a document [36].

Table 2 shows sample sentences included in the summary created by our summarizer and each one of the baseline extractive summarization systems, respectively using the reviews for the user query, "The positives and negatives of Ford F-150 diesel car".

4.3.2 Evaluations Based on Various ROUGE Scores

One of the most commonly-used metrics to automatically evaluate opinion summaries is the **ROUGE**, which is a metric developed for performing text summarization evaluation. ROUGE, which stands for Recall-Oriented Understudy for Gisting Evaluation, is essentially a set of metrics for evaluating an automatic summarization of texts, called a *candidate summary*, against a set of human-created summaries, called *reference summaries*. ROUGE counts the overlap of word or word units between a candidate summary and a set of reference summaries, and is assigned a scalar value in the range [0, 1]. A ROUGE score that is close to zero indicates that the similarity between the candidate and references is low, whereas a ROUGE score close to one indicates the degree of similarity between candidate and references is high. Different variants of ROUGE have been proposed, including ROUGE-N, ROUGE-L, ROUGE-S, and ROUGE-SU. These ROUGE values are determined by counting the number of overlapping units of n-gram, word sequences, and word pairs between the candidate summary and the reference summaries, respectively.

ROUGE-N. Given a set of reference summaries and a candidate summary, the recall ROUGE-N is computed as follows [23], whereas the precision ROUGE-N can be defined accordingly with the denominator replaced by the total count of all the n-grams in the candidate summary:

$$ROUGE\text{-}N = \frac{\sum_{S \in Refereces} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in Refereces} \sum_{gram_n \in S} Count(gram_n)} \quad (18)$$

where $Count_{match}(gram_n)$ denotes the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

ROUGE-L. ROUGE-L measures the *longest common subsequence* (LCS) between a candidate summary and reference summaries. It counts the number of *longest sequence of tokens* that is shared between the candidate summary and a reference summary and identifies longest co-occurring in sequence n-grams automatically. Intuitively, the longer the LCS of two summary sentences is, the more similar the two summaries are. This measure considers the sentence level structure similarity and identifies longest co-occurring in sequence of n-grams. ROUGE-L is formally defined [23] as

$$\begin{aligned} ROUGE\text{-}L_{Recall} &= \frac{LCS(Reference, Candidate)}{M} \\ ROUGE\text{-}L_{Precision} &= \frac{LCS(Reference, Candidate)}{N} \end{aligned} \quad (19)$$

where M (N, respectively) denotes the length of the reference (candidate, respectively) summary.

ROUGE-S. Skip-bigram, a co-occurrence statistics method, considers any pair of words in their sentence order and matches two non-contiguous words with other words in between. It measures the overlap of skip-bigrams between a candidate summary and a set of reference summaries. ROUGE-S2, a special case of ROUGE-S, is the measure of skip-bigram with 2 gaps allowed. Given a reference summary X and a candidate summary Y, the ROUGE-S (skip-bigram) is defined as follows [23]:

$$\begin{aligned} ROUGE\text{-}S_{Recall} &= \frac{Skip_2(X,Y)}{C(M,2)} \\ ROUGE\text{-}S_{Precision} &= \frac{Skip_2(X,Y)}{C(N,2)} \end{aligned} \quad (20)$$

where M and N are as defined in Equation 19, $Skip_2(X,Y)$ denotes the number of skip-bigram matches between X and Y, and C stands for *combination* in mathematics.

ROUGE-SU. SU in ROUGE-SU denotes Skip-Unigram. ROUGE-SU, which is an extension of ROUGE-S, allows the addition of unigram as a counting unit so that it determines the Skip-bigram plus unigram-based co-occurrence statistics. With that in mind, Rouge-SU is like a normal unigram but allows any number of gaps between any two words in sentence order. ROUGE-SU4, a special case of ROUGE-SU, refers to skip unigrams with 4 gaps allowed.

4.3.3 Evaluations Based on Unigram Precision, Recall, and F-measure

The performance of our summarizer and the extractive summarization systems as presented in Section 4.3.1 is first evaluated using the performance measures of ROUGE-1 precision, recall, and F-measure [43]. *Precision* computes the ratio of sentences included in a summary that are relevant, *recall* determines the fraction of relevant sentences that are contained in the summary, whereas *F-Measure* is the harmonic mean of the *precision* and *recall* measures as defined below.

[19] Significant factor of a sentence is defined in Equation 6 based on *significant words* as presented in Section 3.3.1.

Table 2: Sample sentences extracted from the summary generated by Latent Semantic Analysis (LSA), KL-Sum, SumBasic, Luhn, and our summarizer, respectively for the user query Q, "The positives and negatives of Ford F-150 diesel car"

Summarizer	Sample Sentences Extracted from the Summary of Query Q
LSA	I suggest driving one and not taking some of the so called experts advice. I wanted an SUV that had the comfort of a nice car and this fit the bill.
KL-Sum	Smooth power smooth handling. It is a safe car with good handling and a good ride. I have also had the Power Take-off Unit PTU replaced in my vehicle.
SubBasic	This is a great car. Open and airy. I averaged 24. I love my Edge. Very comfortable to ride in from all seats. Performance of the 3.
Luhn	This CUV is excellent looking and very nice. We had an Expedition before so we decide to go for better gas mileage. It is a safe car with good handling and a good ride.
Ours	I get great gas mileage 24 mpg hwy and the car is very easy to drive and fairly responsive for a SUV. I am an American and drive an American icon and couldn't be more proud.

$$
\begin{aligned}
&\textit{Precision} = \\
&\qquad \frac{|\{\textit{Retrieved Sentences}\} \cap \{\textit{Relevant Sentences}\}|}{|\{\textit{Retrieved Sentences}\}|} \\
&\textit{Recall} = \frac{|\{\textit{Retrieved Sentences}\} \cap \{\textit{Relevant Sentences}\}|}{|\{\textit{Relevant Sentences}\}|} \\
&\textit{F-measure} = 2 \times \frac{\textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}}
\end{aligned} \tag{21}
$$

Figure 4 depicts the performance evaluation on our extractive summarizer and the four baseline extractive summarization systems based on ROUGE-1 precision, recall, and F-measure using four different user queries as test cases. According to the computed ROUGE scores, the evaluation shows that the summaries created by our summarizer measure up to our expectation, since they outscore each one of the four baseline summarizers based on the corresponding extractive summary generated for each test query.

4.3.4 Evaluations Based on ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-S, and ROUGE-SU

Besides showing the ROUGE-1 precision, recall, and F-Measure values for a few summaries created by our summarizer and the four baseline models as presented in Section 4.3.3, we have conducted a thorough empirical study on the performance of our summarizers, as well as the four baseline systems, using a set of 270 user queries created by 56 Facebook users between May 2, 2022 and May 12, 2022. These user queries were solicited among different friends of the authors and they were on different products, including motorcycles, cars, hotel lodgings, and housing. Different ROUGE scores, as presented in Section 4.3.2, were computed using the summaries created by our summarizer, Luhn's algorithm, KL-Sum, Latent Semantic Analysis (LSA), and SumBasic, and the results are shown in Table 3. The results are statistically significant based on the Wilcoxon Signed-Ranks Test ($p < 0.03$). This empirical study verifies that our summarizer has outperformed the four baseline extractive summarization approaches that are well-known summarization methods.

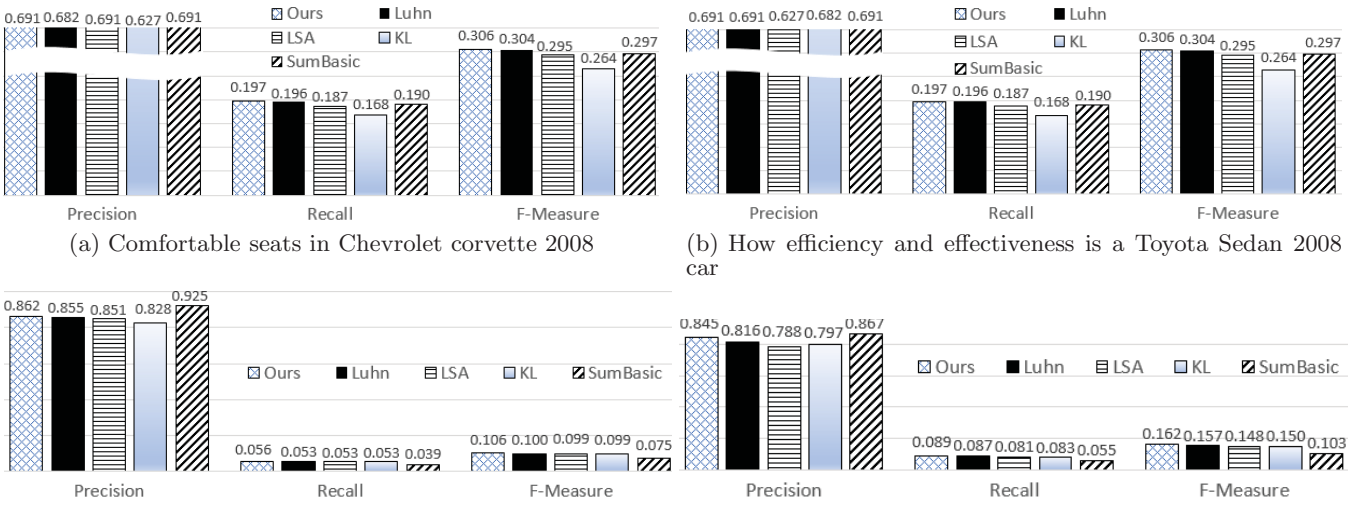
We computed all the ROUGE-1, ROUGE-2, and ROUGE-L scores using the Python implementation made available at the GitHub Repo website[20], whereas all the ROUGE-S and ROUGE-SU scores were calculated using the Java implementation posted under the GitHub Repo website[21].

4.3.5 Linguistic Quality Measures of Extractive Summaries

To further evaluate the quality of the multi-document summaries created by our extractive summarizer and compare its performance with other extractive summarization approaches, we apply the *quality measures* defined in the literature [9, 42] that assess the quality of summaries generated by extractive summarization models. These linguistic quality measures reflect the overall quality of a generated summary in terms of its *grammaticality*, *non-redundancy*, *structure and coherence*, *focus*, and *readability*. Each of the five different measures is computed for each multi-document summary created by the summarizer to be evaluated.

- *Grammaticality*: A (high-quality) summary should not exhibit any *system-internal formatting* (e.g., html formatting tags), *capitalization errors*, or *grammatical mistakes* in sentences (e.g., fragments and missing components) that cause the text to be difficult to read.
- *Non-redundancy*: A summary should avoid unnecessary repetition, such as complete sentences that are repeatedly shown in a text, replicated facts, and repeated use of noun phrases.
- *Structure and Coherence*: A summary should be well-structured and organized, which should not be a heap of related information, but constructed from sentences that yield a *coherent* body of information on a specific topic. It should be easy to identify to whom or what each pronoun/noun phrase refers. If a product is mentioned, its role should be clear. A reference is *unclear* if a product is referred but its identity or relation to the remaining content is *unknown*.
- *Focus*: A summary should have a focus such that sentences in the summary should only contain information

[20] https://github.com/google-research/google-research/tree/master/rouge
[21] https://github.com/kavgan/ROUGE-2.0



(a) Comfortable seats in Chevrolet corvette 2008

(b) How efficiency and effectiveness is a Toyota Sedan 2008 car

(c) Is BMW 2005 car good or bad

(d) Does Audi A4 come with a good engine and price

Figure 4: Performance comparisons on the ROUGE-1 Precision, Recall, and F-Measure values achieved by various extractive summarization systems, including our summarizer, based on four different test cases, i.e., user queries on products

that is related to the rest of the summary. As stated in [21, 33], a focused output of a summary should have related semantics between adjacent sentences.

- *Readability*: A summary should be *easy to read* and its content should be *easy to understand*. The *readability* score of a summary S is a score determined by the grammaticality, non-redundancy, structure, coherence, and focus of the text in S. The readability score is computed by averaging the four other measures that have been normalized so that none of the four linguistic quality measures is weighted heavier than the others to achieve a good balance in terms of treating each linguistic measure equally important.

To establish a baseline measure on the *quality scores* achieved by our summarizer, we (i) used the same set of user queries and dataset for computing various ROUGE scores and (ii) based on the linguistic quality scores listed above to compare the performance of the four multi-document extractive summarization models presented in Section 4.3.1, i.e., *Luhn, KL-Sum, LSA, and SumBasic*. As shown in Table 4, our summarizer is *significantly outperformed* (with 95% confidence) by *none* of the four baseline summarizers in terms of creating summaries that are lower in any of the five quality measures, i.e., grammaticality, non-redundancy, structure and coherence, focus, or readability[22]. Our summarizer achieves the highest *Readability* score among all the four baseline summarizers, in addition to outperform at least one baseline model for each one of the other four linguistic measures. (See the detailed comparison of the performance measures of our summarizer with the four baseline models for each of the linguistic quality measures in Figure 5.) We implemented all of the linguistic quality measures based on the proposed metrics presented in [42].

[22]Note that all of the quality measures were computed using the complement of each measure, i.e., subtracting the original score achieved by each measure from 1.0.

5. CONCLUSION

To facilitate the task of synthesizing opinions expressed in user reviews on a particular product specified in a user query/ question, we have proposed a multi-document sentiment summarization system that is unique in terms of its design. Most prominently, our design has an effective heuristic-based sentence selection process which *retains* sentiment polarity and essential facets in the resultant summary while minimizes *redundancy* and maximizes the *coverage of information* from various information sources. The relatively straightforward approach of our proposed summarizer enhances the current design on summarization with its *effectiveness* and *simplicity*. The effectiveness allows its users to extract desired information without missing essential ideas captured in user reviews. The simplicity of our summarization approach (i) avoids the application of complex natural language processing and machine learning algorithms that require comprehensive training for performing the summarization task, and (ii) can be generalized to any documents, including news articles. Conducted empirical studies on TAC-08, ROUGE scores, and linguistic quality measures also verify that our summarizer ranks near the top among the state-of-the-art approaches in generating query-based, high-quality, sentiment summaries that satisfy the information needs specified in users' queries.

6. REFERENCES

[1] A. Aizawa. An Information-Theoretic Perspective of TF–IDF Measures. *Information Processing & Management*, 39(1):45–65, 2003.

[2] A. Aker, M. Paramita, E. Kurtic, A. Funk, E. Barker, M. Hepple, and R. Gaizauskas. *Automatic Label Generation for News Comment Clusters*. Association for Computational Linguistics, September 2016.

[3] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. Trippe, J. Gutierrez, and K. Kochut. Text

Table 3: Different ROUGE scores achieved by various extractive summarization approaches based on a set of test queries

Average ROUGE/Model	Ours	Luhn	KL-Sum	LSA	SumBasic
ROUGE-1 Recall	0.177	0.175	0.172	0.166	0.156
ROUGE-1 Precision	0.698	0.693	0.684	0.661	0.710
ROUGE-1 F-Measure	**0.282**	**0.279**	**0.275**	**0.266**	**0.256**
ROUGE-2 Recall	0.038	0.036	0.033	0.029	0.028
ROUGE-2 Precision	0.201	0.198	0.184	0.152	0.196
ROUGE-2 F-Measure	**0.064**	**0.061**	**0.055**	**0.049**	**0.050**
ROUGE-L Recall	0.076	0.075	0.072	0.070	0.067
ROUGE-L Precision	0.330	0.329	0.320	0.305	0.349
ROUGE-L F-Measure	**0.124**	**0.122**	**0.118**	**0.114**	**0.112**
ROUGE-S Recall	0.041	0.039	0.033	0.031	0.032
ROUGE-S Precision	0.205	0.204	0.185	0.155	0.235
ROUGE-S F-Measure	**0.068**	**0.065**	**0.056**	**0.052**	**0.056**
ROUGE-SU Recall	0.075	0.074	0.068	0.066	0.063
ROUGE-SU Precision	0.316	0.315	0.308	0.272	0.359
ROUGE-SU F-Measure	**0.108**	**0.107**	**0.095**	**0.094**	**0.090**

Table 4: Comparisons between Ours, Luhn, KL(-Sum), Latent Semantic Analysis (LSA), and SumBasic (SB) summarizers based on the linguistic quality measures

Our Summarizer	Out-Performed by	Outperforms	Significantly Out-Performed by	Significantly Outperforms
Grammaticality	SB	Luhn, LSA	**None**	**Luhn, LSA**
Non-redundancy	None	KL, SB	**None**	**KL, SB**
Structure & Coherence	Luhn, KL, LSA	SB	**None**	**SB**
Focus	None	KL, SB	**None**	**KL, SB**
Readability	None	Luhn, KL, LSA, SB	**None**	**Luhn, KL, LSA, SB**

Summarization Techniques: A Brief Survey. *arXiv preprint arXiv:1707.02268*, 2017.

[4] S. Bahrainian and A. Dengel. Sentiment Analysis and Summarization of Twitter Data. In *IEEE CSE*, pages 227–234, 2013.

[5] M. Bidoki, M. Moosavi, and M. Fakhrahmad. A Semantic Approach to Extractive Multi-Document Summarization: Applying Sentence Expansion for Tuning of Conceptual Densities. *Information Processing & Management*, 57(6):102341, 2020.

[6] R. Blair and J. Higgins. Comparison of the Power of the Paired Samples T-test to that of Wilcoxon's Signed-Ranks Test under Various Population Shapes. *Psychological Bulletin*, 97(1):119, 1985.

[7] S. Chevalier. *U.S. Online Shopper Online Reviews Consumption by Product Type*. Consumer Research Report, 2021.

[8] W. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 2010.

[9] H. Dang. Overview of DUC 2006. pages 1–10, 2006.

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2018. arXiv:1810.04805.

[11] D. Dunlavy, D. O'Leary, J. Conroy, and J. Schlesinger. QCS: A system for querying, clustering and summarizing documents. *IPM*, 43(6):1588–1605, 2007.

[12] U. Farooq, H. Mansoor, A. Nongaillard, Y. Ouzrout, and M. Qadir. Negation Handling in Sentiment Analysis at Sentence Level. *Computer*, 12(5):470–478, 2016.

[13] K. Ganesan and C. Zhai. Opinion-Based Entity Ranking. *Information Retrieval*, 15:116–150, 2012.

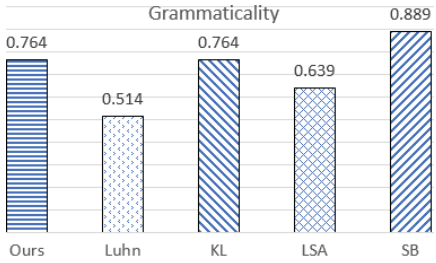
[14] Y. Gong and X. Liu. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In *Proceedings of the* 24^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19–25, 2001.

[15] A. Haghighi and L. Vanderwende. Exploring Content Models for Multi-Document Summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, 2009.

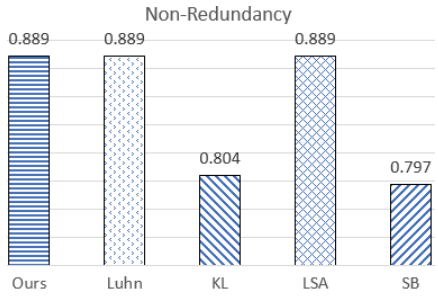
[16] J. He, P. Duboue, and J. Nie. Bridging the Gap between Intrinsic and Perceived Relevance in Snippet Generation. In *COLING*, pages 1129–1146, 2012.

[17] Y. Huang, Z. Liu, and Y. Chen. Query Biased Snippet Generation in XML Search. In *ACM SIGMOD*, pages 315–326, 2008.

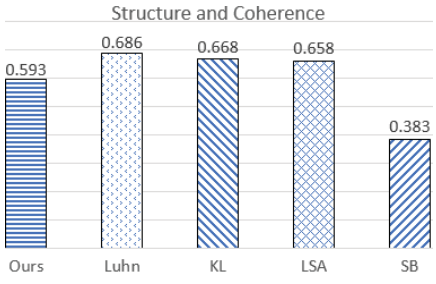
[18] D. Inouye and J. Kalita. Comparing Twitter



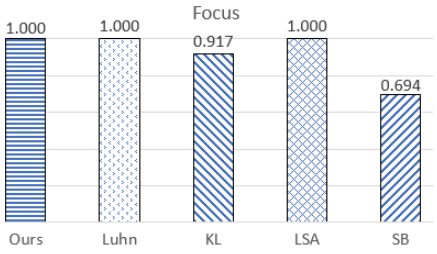
(a) Linguistic Quality: Grammaticality



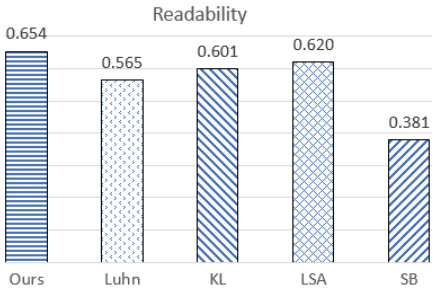
(b) Linguistic Quality: Non-Redundancy



(c) Linguistic Quality: Structure & Coherence



(d) Linguistic Quality: Focus



(e) Linguistic Quality: Readability

Figure 5: Performance comparison on the linguistic quality measures of various extractive summarization models, in additional to our summarizer

Summarization Algorithms for Multiple Post Summaries. In *2011 IEEE Third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pages 298–306. IEEE, 2011.

[19] H. Jeong, Y. Ko, and J. Seo. Efficient keyword extraction and text summarization for reading articles on smart phone. *Computing & Informatics*, 34(4):779–794, 2015.

[20] P. Judea. *Probabilistic Reasoning in the Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[21] A. Knott, J. Oberlander, M. O'Donnell, and C. Mellish. Beyond Elaboration: The Interaction of Relations and Focus in Coherent text. *Text Representation: Linguistic and Psycholinguistic Aspects*, pages 181–196, 2001.

[22] S. Lamsiyah, A. E. Mahdaouy, B. Espinasse, and S. Ouatik. An Unsupervised Method for Extractive Multi-Document Summarization Based on Centroid Approach and Sentence Embeddings. *Expert Systems with Applications*, 167:114–152, 2021.

[23] C. Lin. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop on Text Summarization Branches Out*, pages 74–81, 2004.

[24] J. Lin and D. Demner-Fushman. Will pyramids built of nuggets topple over? In *HLT/NAACL*, pages 383–390, 2006.

[25] Y. Liu and Y. Zheng. One-against-all multi-class svm classification using reliability measures. In *IJCNN*, pages 849–854, 2005.

[26] G. Luger. *Artificial Intelligence: Structures & Strategies for Complex Problem Solving,* 6^{th} *Ed.* Addison Wesley, 2009.

[27] Y. Meena and D. Gopalani. Analysis of Sentence Scoring Methods for Extractive Automatic Text Summarization. In *Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*, pages 1–6, 2014.

[28] B. Mutlu, E. Sezer, and M. Akcayol. Multi-Document Extractive Text Summarization: A Comparative Assessment on Features. *Knowledge-Based Systems*, 183:104848, 2019.

[29] R. Nallapati, F. Zhai, and B. Zhou. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. In *AAAI*, pages 3075–3081, 2017.

[30] A. Nenkova and L. Vanderwende. The Impact of Frequency on Summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005*, 101, 2005.

[31] J. Ni, J. Li, and J. McAuley. Justifying Recommendations Using Distantly-Labeled Reviews and Fine-Grained Aspects. In *EMNLP-IJCNLP*, pages 188–197, 2019.

[32] S. Osinski. Improving Quality of Search Results Clustering with Approximate Matrix Factorisations.

In *ECIR*, pages 167–178, 2006.

[33] E. Pitler, A. Louis, and A. Nenkova. Automatic Evaluation of Linguistic Quality in Multi-Document Summarization. In *Proceedings of the* 48^{th} *Annual Meeting of the Association for Computational Linguistics*, pages 544–554, 2010.

[34] J. Sanchez-Gomez, M. Vega-Rodríguez, and C. Pérez. Extractive Multi-Document Text Summarization Using a Multi-Objective Artificial Bee Colony Optimization Approach. *Knowledge-Based Systems*, 159:1–8, 2018.

[35] B. Schiffman, A. Nenkova, and K. McKeown. Experiments in multi-document summarization. In *HLT*, pages 52–58, 2002.

[36] D. Shen, Z. Chen, Q. Yang, H. Zeng, B. Zhang, Y. Lu, and W. Ma. Web-Page Classification through Summarization. In *Proceedings of the* 27^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 242–249, 2004.

[37] J. Steinberger, M. Poesio, M. Kabadjov, and K. Ježek. Two Uses of Anaphora Resolution in Summarization. *Information Processing & Management*, 43(6):1663–1680, 2007.

[38] J. Tohalino and D. Amancio. Extractive Multi-Document Summarization Using Multilayer Networks. *Physica A: Statistical Mechanics and its Applications*, 503:526–539, 2018.

[39] S. Trauzettel-Klosinski and K. Dietz. Standardized Assessment of Reading Performance: The New International Reading Speed Texts IreST. *Investigative Ophthalmology & Visual Science*, 53:5452–5461, 2012.

[40] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova. Beyond SumBasic: Task-Focused Summarization with Sentence Simplification and Lexical Expansion. *Information Processing & Management*, 43(6):1606–1618, 2007.

[41] M. Woolfe. *A Statistical Analysis of 1.2 Million Amazon Reviews*. Minimaxir.com, June 2014.

[42] W. Zhu and S. Bhat. Gruen for Evaluating Linguistic Quality of Generated Text. *arXiv preprint arXiv:2010.02498*, 2020.

[43] L. Zhuang, F. Jing, and X. Zhu. Movie Review Mining and Summarization. In *Proceedings of the* 15^{th} *ACM International Conference on Information and Knowledge Management (CIKM)*, pages 43–50, 2006.