Summarization Assessment Methodology for Multiple Corpora Using Queries and Classification for Functional Evaluation

Sam Wolyn[a] and Steven J. Simske[b]
[a] *Systems Engineering Department, 6209 Campus Delivery, Colorado State University, Fort Collins, CO, USA 80523*
[b] *Systems Engineering Department, 6209 Campus Delivery, Colorado State University, Fort Collins, CO, USA 80523*

Abstract. Extractive summarization is an important natural language processing approach used for document compression, improved reading comprehension, key phrase extraction, indexing, query set generation, and other analytics approaches. Extractive summarization has specific advantages over abstractive summarization in that it preserves style, specific text elements, and compound phrases that might be more directly associated with the text. In this article, the relative effectiveness of extractive summarization is considered on two widely different corpora: (1) a set of works of fiction (100 total, mainly novels) available from Project Gutenberg, and (2) a large set of news articles (3000) for which a ground truthed summarization (gold standard) is provided by the authors of the news articles. Both sets were evaluated using 5 different Python Sumy algorithms and compared to randomly-generated summarizations quantitatively. Two functional approaches to assessing the efficacy of summarization using a query set on both the original documents and their summaries, and using document classification on a 12-class set to compare among different summarization approaches, are introduced. The results, unsurprisingly, show considerable differences consistent with the different nature of these two data sets. The LSA and Luhn summarization approaches were most effective on the database of fiction, while all five summarization approaches were similarly effective on the database of articles. Overall, the Luhn approach was deemed the most generally relevant among those tested.

Keywords: Abstractive summarization, analytics, compression, extractive summarization, machine learning, phrases, repurposing, saliency, sentence, statistical learning, style

1. Introduction

Summarization is an important element in many computer-aided text analytics tasks, including indexing, key word generation, translation, clustering, classification, and curriculum generation (document sequencing, or reading order). Summarization approaches, at the broadest level, can be described as either extractive or abstractive. Extractive summarization consists of selecting a subset of the text phrases in the original document. This partial selection is thus a lossless compression of the content, in the sense that the text selected is unaltered from its original form. Abstractive summarization, in contrast, is a form of lossy compression since the text is altered semantically from the original, usually incorporating novel words, word order, and/or phrases in order to provide a plainspoken summary [1]. This may be in a different voice altogether, for example in a neutral conversational style. An abstractive summary has the advantage of being semantically edited so that the selected phrases may flow together better than an extractive summary. However, an abstractive summary may not represent well, if at all, the style, tone, or voice of the original author. The summary may, additionally, introduce a different vocabulary than the author themselves used, with potentially deleterious effects on subsequent text analytics including indexing, clustering, and categorization.

In 1958, Luhn noted that machine learning methods could be used to automatically create abstracts for technical literature that were not highly influ-

enced by the unintentional attitudes, backgrounds, biases, and opinions of the human ground truthing normally employed for cataloguing purposes [2]. Many automatic summarization techniques continue to be based on Luhn's approach of using word-frequency and contextual cues. The idea of assigning measure (or weights) of significance to each of the sentences is still employed today. Other early works include those by Edmundson and Wyllys [3,4], Rush et al. [5], Sparck-Jones [6], and Kupiec [7].

The first reported extractive approach [2] defined a word frequency-based measure using the idea that the more often an important word appears in a phrase, sentence, or other unit of text, the more significant the sentence. This increases its likelihood of being in the summary. Word frequency, however, was shown to require normalization by the document length, creating a more broadly applicable weighting factor for significance.[8] TF*IDF, or Term Frequency times Inverse Document Frequency, is a related approach of importance to extractive summarization. In addition to frequency-based approaches, there are several feature-based approaches that assume that certain characteristics of text identify significant sentences or phrases. For example, titles, abstracts, sentence position, proper nouns, and the like are indicators of relative importance [8]. Such approaches may require additional methods such as document segmentation in order to identify these features.

Other extractive techniques have been historically based on a wide variety of now-familiar machine learning approaches. Naïve Bayes classifiers [8], support vector machines (SVM) [9], and artificial neural networks (ANN) [10] have been employed. Bayes classifiers are employed to assign the probability of a sentence being included in the summary based on a set of sentence features, wherein the highest ranked sentences are chosen [8]. SVM approaches are used for a series of binary classifications; for example, associated with a decision tree. In one ANN approach, summary sentences were themselves data mined for features that best characterize them [10]. In another, a set consisting of Cable Network News (CNN) articles was used to train an ANN to rank sentences, with the summaries consisting of the highest ranked sentences [11]. A larger set of CNN articles was used, together with the author-provided summaries, to provide a useful large training set for extraction summarization [12]. Recurrent neural networks have also been recently deployed for extractive summaries [13]. These are other recent advances in extractive summarization are overviewed in a recent survey [14]. Some of the limitations on summarization, particularly from the lens of compressibility, has also been recently provided [15].

In generalized extractive summarization, text elements (phrases, sentences, etc.) are selected based on P different characteristics, among which are sentence scoring, cue phrases, sentence inclusion of numerical data, sentence length, sentence position, sentence centrality, sentence resemblance to the title, and graph scoring. The total sentence score, or TSS (Eq. 1), is the sum for all N terms of the product of the weighting factors (WF) of each of the P characteristics (these default to 1.0 and vary from 1.0 based on the relevance of each word in the overall document.

$$TSS = \sum_{i=1}^{N} \prod_{k=1}^{\mathrm{P}} WF_i(k) \qquad (1)$$

The problem with the approach of Eq. 1 is that similar sentences that represent the most relevant terms may all be highly scored, leading to an initial summary (either extractive or the source for abstraction) that does not represent the range of content in the document. In order to prevent such replication, one approach is selectively down-weight terms once they have been selected, as noted in Eq. 2. [16]

$$RSS = \sum_{i=1}^{N} \prod_{k=1}^{\mathrm{P}} WF_i(k) - \lambda \sum_{i=1}^{N} WF_i(k)\, num_i(S) \qquad (2)$$

In Eq. 2, the Regularized Sentence Score, or RSS, adjusts the TSS by multiplying a regularization factor, λ, with the sum of occurrences of each of the terms in any sentences that have already been assigned to the summary. This is termed $num_i(S)$ for each of the N terms in the sentence being regularized, as they occur in the set of sentences already part of the summary. In this way, each sentence score is updated after the summary is appended. The regularization sum (right side of Eq. 2) penalizes selecting more sentences highly similar to the ones already part of the summary.

Abstractive approaches, on the other hand, do not necessarily use the source text unaltered, but instead usually paraphrase the text or transform the text to build a summary. The abstractive approach generally incorporates a larger initial subset of the document than the extractive summary, though it may directly transform the extractive summary. Abstractive approaches can result in the production of inac-

curate details and the repetition of information. Both of these issues are addressed by a "Pointer-Generator Network" [17], which is built on top of a "sequence-to-sequence" model. This is a now common abstractive summarization approach employing recurrent neural networks (RNNs). The Pointer-Generator Network copies words from the source text (to address the inaccurate details) by pointing and continuity-providing words are produced using the generator. They also track the summarization as it is being produced in order to avoid repetition. Another abstractive summarization approach is the use of an Attentional Encoder-Decoder Recurrent Neural Network [18]. Using the decoder to "point" (copy) a word or the encoder to "generate" (produce) a word addresses the inaccurate detail concerns. The summary outputs from the Encoder-Decoder system tended to contain repetitive phrases, prompting the researchers to add a Temporal Attention Model that essentially keeps track of which parts of the document it has already processed and discourages it from looking at those pieces again. A more comprehensive method of preventing the summary from being overwhelmed with repetitive content is described elsewhere [16].

Another approach to abstractive summarization uses a different approach, called Attention-Based Summarization, which joins an ANN model with an attention-based encoder. The language model consists of a feed-forward neural network language model (NNLM) used for estimating the contextual probability of the next word, while the encoder acts as a conditional summarization model [19]. In general, each word of a summary is generated based on the input sentence. The goal of the experiments is to use the summarization model for generating headlines. Training occurs by pairing a headline with the first sentence of an article (rather than the entire article) to create an input-summary pair. The extractive tuning addresses the issue of inaccurate details (similar to decoding or pointing) by tuning with a small number of parameters after the model is trained.

The summarization approaches in the articles reviewed here, together with the summarization experiments covered in the survey articles, focus on individual corpora for summarization analysis. This paper applies multiple summarization approaches to two *fundamentally different* corpora. In so doing, the differences in summarization approaches that are germane to providing downstream utility (including indexing, clustering, classification, and reading order) of the summarized texts are compared. The primary motivation of the research in this article is to provide two distinct mechanisms for grading the summaries provided by multiple summarization algorithms. The first is performed using a query set, in which the functionally optimum summarization algorithm is determined to be the one in which query behavior is most similar in comparing the original documents to the summarized documents. This is particularly suitable to longer documents, and is illustrated herein using longer literary works. The second mechanism for grading summaries is performed using document classification. Here, the functionally optimum summarization approach is determined to be the one resulting in the best classification accuracy for the documents. This is particularly suited to articles, and is applied to a well-established set of CNN articles available to researchers.

In Section 2, the methods and experimental designs are described. Specifically, the query-based and classification-based methodologies for evaluating summarization output are introduced. In Section 3, an overview of the primary results is given. In Section 4, the results are evaluated more thoroughly in a discussion, and concluded with potential follow-on work.

2. Methods and Experimental Designs

2.1. Document Corpora, the Use of Query to Rank Summarizers, and Query Sets

In order to begin exploring the differences in summarization for different types of document corpora, two widely different, publicly available large corpora of documents, were selected. The first set of documents consisted of 100 works of fiction (novels) with lengths ranging from 17,074 to 562,199 words with a mean of 125,565 words selected from Project Gutenberg [20, 21]. All of the works selected were in English, and all metadata was removed from the document, except for chapter and part labels. Introductions were also removed, even if they were written by the original author. Footnotes were also removed. The texts were also reformatted, removing any newlines or indents used for formatting.

The second set of documents consisted of the CNN Corpus [12]. The corpus consists of 3000 news articles covering 12 different subjects, including business, health, and politics. The corpus was chosen because it contains a human-selected gold standard for summarization. For most articles, the gold standard is three sentences, provided as a leader for the article on the cnn.com website. All of the articles were also in English.

Summaries of the documents (novels and articles) were generated by Sumy, a Python library that performs various summarization algorithms on texts of different languages. It has a built-in parser and tokenizer, so that minimal preprocessing had to be done on the documents. Five different algorithms within Sumy were used to summarize the datasets: LexRank, LSA, Luhn, Reduction, and Sum Basic. There were three other summarization algorithms in Sumy: Edmundson, KL, and TextRank. Edmundson was not used as it required the definition of "bonus words" (more appropriate for search query investigations), while KL and TextRank suffered poor processing times for summarizing the longer documents. As a baseline, the documents were also summarized several times using Sumy's Random summarizer, which as the name implies selected sentences randomly from the documents. The novels had summaries generated comprising 20, 50, and 100 sentences, while the much shorter articles summaries generated comprising 3, 5, and 10 sentences.

The Sumy summarizers are described briefly here. The Random summarizer just selects sentences at random from the documents, and is a point of comparison, with only poor summarizers approaching the performance of the Random one. The Luhn summarizer uses a simple "significant word" algorithm to select sentences for the summaries. The significant words occur with high frequency in the text, but are not stop words. It can be thought of as a TF (term frequency) approach. The Latent Semantic Analysis, or LSA, method identifies synonyms in the text and topics that are not explicitly stated in the text. The LexRank summarizer is another unsupervised approach that discovers connections between the sentences and selects for the summaries those that are connected with the most significant words/topics. The SumBasic is a sentence score summing approach, which is generally considered a baseline "reasonable" summarizer, which any advanced summarization approach should surpass. Finally, the Reduction algorithm is a graph-based summarization, where sentences are weighted by the sum of the weights of its edges to other sentences, with weight computed in a manner similar to LexRank.

After performing each summarization, a functional approach (that is, quantitative but not requiring human ground truthing) to assess the efficacy of summarization was devised. In order to provide this, the generation of a query set to use for searching both the original documents and their summaries was required. For each query, a search process is performed where an ordered (ranked) list of documents is returned, with the best match having the highest rank in the search. In theory, a perfect summarizer will result in the same query behavior on the summarized set as occurred for the original document set. One goal of this paper will be to explore whether the query matching does in fact correlate with summarization accuracy.

The query set for the documents was generated after first noting the themes provided by Sparknotes for the novels. The themes provided words relevant to the texts while also not being pulled directly from the text (i.e. abstractive rather than extractive). 303 words were taken from the themes and used as an initial query set. These were considered both relevant and likely to be broadly applicable (that is, applicable to the article set as well as the document set). The query set was then expanded by adding common words, common animals, common colors, and common countries to generate a new set of 403 query words. The common words were taken from a list of 100 common words, and duplicates with words generated from the themes were removed. 10 common animals, such as "cat" and "dog" were used, as well as 10 colors, which consisted of primary and secondary colors, as well as black, white, gray, and brown. The countries were chosen by selecting the 30 most populous countries in the world in the 1800s, as most books were written before 1900 in order to be in public domain when the experiments were performed in 2021. The same query set was used throughout, and every document returned a query score for each query as described in the next section.

2.2. Scoring Approach

For each query word and document, a score was generated. For each word in the document, if the word exactly matched the query word, it was given a score of 1. If it shared a lemma with the query word, it was given a score of 0.95. If the words were synonyms according to Wordnet, the word would be given a score of 0.75. The results were not particularly dependent on these scoring methods; for example, if lemmas were given scores of 1.0 and synonyms were given scores of 0.6-0.9, the outcomes were relatively similar. If the word did not match any of the 403 query terms, it was given a score of 0.0. The scores for each word were added together, and this sum was divided by N, the number of words in the document (Eq. 3), to yield the overall score of the document for each query term.

$$Score = \frac{1}{N} * \sum_{Word\ in\ document} Score\ of\ word \quad (3)$$

Once all the scores were generated, the documents were sorted for each query word based on the calculated score. Eq. 4 is used to derive the Total Query Error (TQE), which is entirely based on the similarity in ranks returned from the query. For example, if Document A was the top scored document for Query 1 for the original texts and was the third best document for Query 1 in the set of summaries in Summary 1, then the error score for one relevant rank would be |3 – 1| = 2. For the result comparing the original documents to a summary, the query error for each query was averaged, producing a single score for the entire comparison. Each error score was calculated using 1, 2, 3, 4, and 5 relevant ranks (rr); that is, by comparing the top 1, 2, 3, 4, and 5 ranked documents from the original document queries. The error score was calculated for both the novels and the articles. In Eq. 4, N_{rr} = the number of relevant ranks, $rank[Q(S)]_{of\ rank[r]\ term\ in\ OD}$ is the rank of the query for the summarized version of the document, and r is the rank of the original document (OD) for the query.

$$QE = \sum_{r=1}^{N_{rr}} (N_{rr} - r + 1) \left(\sum_{n=1}^{N_{Queries}} |rank[Q(S)]_{of\ rank[r]\ term\ in\ OD} - r|\right) \quad (4)$$

An example employing Eq. 4 will be illustrative. Suppose that a specific query term (Q) is run against the original documents (OD), and then separately run against the summaries (S) of these same documents. When Q_on is used on the Original Documents, this is Q(OD), and Documents {A,B,C,D,E,…} are returned in order. Next, the same specific query term Q is run against the summaries, called Q(S). Instead of the order for Q(OD), suppose summaries {C,B,D,H,A,…} are received in this order. The difference in ranked order for the most relevant—that is, ranked first in response to Q(OD)—document, which is A, is rank [1] in Q(OD) and rank [5] in Q(S). The error is therefore |[1]-[5]| = 4. In Eq. 4, then, $rank[Q(S)]_{of\ rank[r]\ term\ in\ OD}$ is 5, and r, the rank in Q(OD), is 1.

The left-hand side of Eq. 4 accounts for the number of relevant ranks. For rank r=1, $N_{rr} - r + 1$ is N_{rr}, or 5 in the case of rr=5. For the rank=2 document B, the multiple on the left of Eq. 4 is 5-2+1 = 4, while the right side is 0 since Q(S)=Q(OD) for B. For the rank=3 document C, the multiple on the left of Eq. 4 is 5-3+1 = 3, while the right side is |[3]-[1]| = 2. The TQE for just these three documents, with rr=5, is 5*4 + 4*0 + 3*2 = 26. Note that if only one rank is relevant (that is, rr=1), the TQE = 4. If rr=2, the TQE = 2*4 + 1*0 = 8. If rr=3, the TQE = 3*4 + 2*0 + 1*2 = 14. It should be noted that these TQE values are for only one query term. The sum of the differences for all query terms (right side of Eq. 4), multiplied by the relative value of each rank (left side of Eq. 4) is reported as the **Error Score**. This Error Score will be the primary set of results in the Results section to follow.

The algorithm for ranking the summaries by their Total Query Error (TQE) in comparing the original documents to their associated summaries is given in Figure 1.

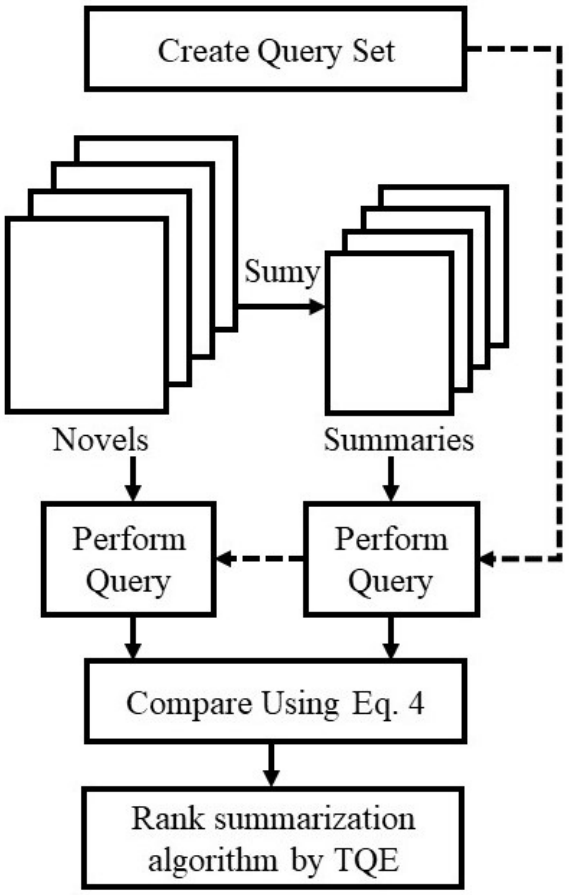

Fig. 1. Query matching functional approach.

The summaries were also used to classify the novels and, separately, the articles. The smallest class size in the articles consisted of 98 articles, so the training and testing sets were set at a size of 49 articles each in order to achieve a balanced 12-class set (that is, training was performed on 50% of the CNN articles in each class, and testing on the remaining 50%). The 50% training was used to balance between under-training and overfitting. The document classification was performed three times, each with a different set of training and testing documents. The test set assignments were randomly generated by sampling each class of article. Each document was classified by taking the frequency of each word in the document and calculating the cosine (Eq. 5) of the document word frequency (the $\vec{a}$ vector in Eq. 5) with the mean word frequencies from training documents from each class (the $\vec{b}$ vector in Eq. 5). The document was assigned to the class it had the largest cosine with.

$$cos(\vec{a},\vec{b}) = \frac{\vec{a} \bullet \vec{b}}{\left|\vec{a}\right|\left|\vec{b}\right|} = \frac{\sum_{w=1}^{N} a_w b_w}{\sqrt{\sum_{w=1}^{N} a_w^2}\sqrt{\sum_{w=1}^{N} b_w^2}} \quad (5)$$

The classifier used an initial TF*IDF (Term Frequency times Inverse Document Frequency) filtering of words for each training set. A wide number (at least 112) variations of TF*IDF exist [22]. The TF*IDF definition employed for a word was calculated by taking the mean number of occurrences of a word in a document in the class, divided by the mean number of occurrences of a word in the documents for all of the classes (Eq. 6). This is the "document frequency", the inverse of which creates the IDF term in TF*IDF. In Eq. 6, "Docs" = the total number of documents in all classes. Because the training set was balanced between classes, this was equivalent to the number of occurrences in a class multiplied by the number of classes, divided by the total number of occurrences. A TF*IDF score of 1 indicated that the word occurred in the class as many times as its mean occurrence in the other classes, and a score of 12 indicated that the word only occurred in that class. The classifier was run on each of the different sets of summarized documents, as well as 10 sets of random summaries at 3, 5, and 10 sentences.

The algorithm for classifying the original CNN articles and the various summaries of the articles is given in Figure 2.

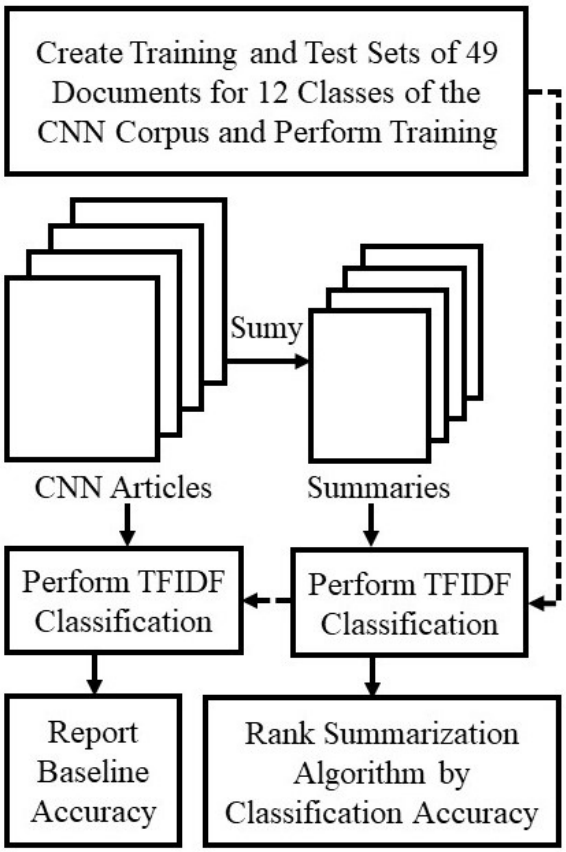

Fig. 2. Classification-based functional approach.

$$TFIDF = \frac{\sum_{Docs\ in\ class} Occurrences\ of\ word\ in\ doc \,/_{Docs\ in\ class}}{\sum_{Docs\ in\ all\ classes} Occurences\ of\ word\ in\ doc \,/_{Docs}} \quad (6)$$

TF*IDF thresholds from 1.0 to 12.0 were used in order to determine the effect of TF*IDF filtering on the classification results for the different types of summaries. For each original text, the TF*IDF of each word was found. The Term Frequency for each word in a summarized text was also found, and the cosine distance was calculated between the original texts and the summarized texts. The summarized document was assigned to the class for which it had the highest cosine similarity. The document frequencies used to calculate the TF*IDF values were normalized, so that the different lengths of the documents did not affect the TF*IDFs. For example, if a word occurred only in a short document, it would have a much higher TF*IDF than a word that only

occurs in a longer document. This normalization is simply representing the percentage of words occurring in the document for each query term, as shown in Eq. 7.

$$TFIDF = \frac{\sum_{Docs\ in\ class} Occurrences\ as\ pct\ words\ in\ doc / Docs\ in\ class}{\sum_{Docs\ in\ all\ classes} Occurences\ as\ pct\ words\ in\ doc / Docs} \quad (7)$$

3. Results

3.1. Summarizing the Novels

For summarizing the novels, only the LSA and Luhn methods outperformed random summarization (Table 1, which provides the Error Score, or mean TQE). Randomly summarized texts were created by randomly selecting sentences from the target text. Because the quality of a randomly generated summary can vary, the documents were randomly sampled 10 times for each summary length. For the 20-sentence summaries, the random summaries' error scores had a mean of 37.90 and a standard deviation of 0.91. For 50-sentence random summaries, the error scores had a mean of 30.55 and a standard deviation of 1.15. Finally, 100-sentence random summaries had error scores with a mean of 23.66 and a standard deviation of 1.77.

For the 20-sentence summaries in Table 1, LSA had an error score 8.8 standard deviations less than the mean for random summaries, and Luhn had an error score 11.5 standard deviations less than the mean for random summaries. For the 50-sentence summaries, LSA error scores were 9.0 standard deviations below those of random summaries, while Luhn error scores were 10.3 standard deviations less. For 100-sentence summaries, LSA and Luhn were 5.7 and 5.3 standard deviations below the scores for random summaries. For 1 relevant rank, a score of 0 denotes that for every query term, the same document has the highest score for the term for both the original and summarized texts. The worst score possible for a single query is the number of documents minus 1, in this case 99, where the top scoring document when looking at the original documents is the lowest scoring when looking at the summarized texts. If the ordering of documents for queries is random, then the mean score is the value of the worst score divided by 2, in this case 49.5.

As the length of the summary increases, the error score for the query words decreases, as the summarized texts have a larger portion of the text from the original texts and are thus more similar to the original documents. This is readily evidenced in Table 1, where for the 20-Sentence Summaries of all 15 approaches the mean Error Score is 37.0; for the 50-Sentence Summaries the mean Error Score is 29.4; and for the 100-Sentence Summaries the mean Error Score is 22.9. However, these Error Scores, even for the Luhn and LSA summarizers, are more than 25% as high as the Error Score for random guessing (49.5) in Table 1, and more than 50% as high as for Random Summarization.

As the number of ranks of relevance (rr) are increased, per Equation 4, the Error Score is a composite of several comparisons. More interesting for the comparisons where rr>1 is the distance of the mean Error Score for the Sumy summarizers in terms of standard deviations from the Random Summarizations. These are provided for the Novels document set in Table 2.

Several trends are consistently observed across Table 2. The first is that LexRank, Reduction, and Sum Basic summarizations do not provide better matching between OD and S document sets than do the *Random* Summarizations. In fact, for LexRank with 20 Sentences; for Reduction with 50 and 100 Sentences; and for Sum Basic across the range of Sentences, these summarization approaches underperform random summarization. The second trend of note is that LSA and Luhn consistently outperform all other summarizations in terms of matching the query responses of the S and OD sets. For all three sets of sentences, and for all five relevant rank values (incorporating the results of Table 1 here), their Error Scores are always more than 7.4 standard deviations less than those of the Random Summarization. The third trend (data shown only for Table 1) is for the Error Score to consistently decrease as the number of sentences. When the six summarizers (including the Random Summarizer as one of the six) are combined in a normalized regression curve, the coefficient of determination (R^2) value for the linear regression of sentences versus Error Score is 0.824, and for a second-order polynomial regression (with some asymptote behavior illustrated), the coefficient of determination was 0.871. This is for rr=1 in Table 1.

Table 1

Error Score (mean of TQE for all queries) for the Novels, 1 Relevant Rank

Approach	**20-Sentence Summary**	**50-Sentence Summary**	**100-Sentence Summary**
LexRank	39.95	30.58	24.60
LSA	29.88	20.26	13.56
Luhn	27.45	18.63	14.29
Reduction	38.51	33.07	27.29
Sum Basic	40.62	33.08	27.02
Random (1)	37.49	31.17	24.86
Random (2)	37.09	32.35	20.50
Random (3)	38.79	31.43	22.11
Random (4)	36.97	30.04	26.59
Random (5)	39.16	30.66	24.64
Random (6)	38.14	29.60	24.49
Random (7)	38.09	30.05	22.59
Random (8)	36.50	30.54	23.29
Random (9)	37.77	28.24	22.60
Random (10)	39.02	31.39	24.94

Table 2

Error Score (mean of TQE for all queries) difference from random, as (mean / standard deviation), Novels, 2-5 Relevant Ranks (rr)

Summary Type	**N Sentences**	**rr=2**	**rr=3**	**rr=4**	**rr=5**
LexRank	20	-2.806	-3.192	-3.255	-3.352
	50	-0.513	-1.362	-2.609	-3.636
	100	-0.493	-0.791	-1.297	-1.820
LSA	20	11.963	12.463	12.110	11.877
	50	12.651	15.131	19.225	21.357
	100	8.024	9.839	12.791	14.299
Luhn	20	14.185	14.082	13.481	13.176
	50	14.512	16.786	21.453	23.670
	100	7.473	9.392	12.660	14.454
Reduction	20	-0.374	0.138	0.511	0.704
	50	-2.024	-2.231	-2.233	-2.254
	100	-2.087	-2.601	-2.843	-2.891
Sum Basic	20	-3.626	-3.498	-3.034	-2.757
	50	-2.892	-4.000	-5.250	-5.846
	100	-2.430	-2.943	-3.798	-4.152

Table 3

Error Score (mean of TQE for all queries) for the CNN articles, 1 Relevant Rank

Approach	**3-Sentence Summary**	**5-Sentence Summary**	**10-Sentence Summary**
LexRank	396.2655	245.2419	118.9715
LSA	469.9479	332.4280	134.0261
Luhn	400.0074	243.8759	117.3449
Reduction	403.5670	299.5620	129.5968
Sum Basic	430.3176	261.2581	141.5062
Random (1)	560.5645	394.9615	152.4057
Random (2)	638.7506	403.9032	206.4206
Random (3)	628.1725	373.0943	177.6774
Random (4)	626.1787	414.8548	188.3573
Random (5)	603.0597	400.3561	188.7543
Random (6)	612.2432	366.2506	200.2047
Random (7)	588.7419	361.2767	160.6241
Random (8)	594.4392	367.0682	168.2382
Random (9)	634.6700	355.3387	180.7320
Random (10)	665.4566	410.1427	159.0484

3.2. Summarizing the Articles

All of the Sumy summarizers provided lower Error Scores for comparing the S and OD sets of articles than the Random Summarizer. The results for Equation 4, using one relevant rank, are presented in Table 3. There, for the 3-sentence random summaries, the average error score was 615.2, with a standard deviation of 29.9. The 5-sentence random summaries had a mean error score of 384.7 and standard deviation 22.3. For the 10-sentence random summaries, the mean error score was 178.2 with a standard deviation of 18.1. For the 3-sentence summaries, all five summarization methods were at least 8 standard deviations away from the random average. For 5-sentence summaries, they were at least 2 standard deviations away, with Luhn being over 6 standard deviations away. For 10-sentence summaries, the summaries were in the range of 2 to 3.5 standard deviations away from the mean of the random summaries.

For the CNN articles, the best possible score remains at 0, but the worst possible score increases to 2999, and the mean score increases to 1499.5, as there are 3000 articles. As with the Novels, the Random Summaries have Error Scores well below simple guessing because they represent a greater percentage of the actual article as the number of sentences increase. For Table 3, it is worth noting that the product TQE * (number of sentences) is relatively constant for all of the summarizers (coefficient of variance for this product has a mean of 0.063). This is decided different behavior than that of the same product for the Novels (Table 1). This more predictable behavior affects the regressions mentioned above. When the six summarizers (including the Random Summarizer as one of the six) are combined in a normalized regression curve, the coefficient of determination (R^2) value for the linear regression of sentences versus Error Score is now a much higher 0.931, and for a second-order polynomial regression (with some asymptote behavior illustrated), the coefficient of determination was 0.988.

As for the Novels, as the number of sentences included in the summary of the CNN articles increases, the error score for the query words decreases. In Table 3, the mean Error Score is 550.2 for 3-Sentence Summaries; 348.6 for the 5-Sentence Summaries; and 161.6 for the 10-Sentence Summaries. Error Scores for all five Sumy summarizers are less than 10% as high as the Error Score for random guessing (1499.5) in Table 3, significantly less than observed for the Novels. However, each of these is more than 50% as high as for Random Summarization, in agreement with the findings for the Novels (Table 1).

As the number of ranks of relevance (rr) are increased, per Equation 4, the Error Score is again a composite of several comparisons. As with Novels, for the CNN articles the mean Error Score across query terms is given in terms of standard deviations from simple, randomized summarizations (Table 4).

In Table 4, each of the Sumy summarizers resulted in Error Score differences from Random Summarization similar to that of the CNN article gold standard set ("Gold Standard" in Table 4). Additionally, all five of the Sumy summarizers significantly outperform the Random Summarizer in terms of matching the query behavior of the summarized (S) and original documents (OD). This is decidedly different behavior than for the Novels, where only two of the Sumy summarizers (LSA and Luhn) outperformed the Random Summarizer by this measurement. Overall, the peak Error Scores are obtained when using "relevant ranks"=4 and three sentences. Here, the peak values are (13.4, 10.0, 14.2, 14.7, and 10.2) for (LexRank, LSA, Luhn, Reduction, and Sum Basic). These compare well with that of the Gold Standard, 10.5.

3.3. Classifying the Articles

Classification of the summarized versions of the articles was next performed. Classification was intended to provide a functional means of assessing the relative value of each summarization approach. The classification approach was based on cosine similarity with TF*IDF thresholded terms, as described above and in previous work [22].

The TF*IDF classifier used achieved as much as 60% accuracy on the testing set, as illustrated in Table 5. Three different trials of training and testing sets were employed, with the mean accuracy being 57% for TF*IDF thresholds from 3.5 to 7.0. The CNN corpus has 12 classes of documents; therefore, 8.3% accuracy is achieved by random guessing.

Table 4

Error Score difference from random, as (mean / standard dev.), Articles, 2-5 Relevant Ranks (rr).

Summary Type	**N Sentences**	**rr=2**	**rr=3**	**rr=4**	**rr=5**
LexRank	3	10.618	13.156	13.390	12.276
	5	8.518	9.453	10.955	10.958
	10	3.768	4.297	5.264	6.113
LSA	3	8.232	10.230	10.020	8.948
	5	5.451	6.771	7.941	8.256
	10	3.248	3.819	4.545	5.628
Luhn	3	12.020	14.441	14.190	12.862
	5	8.578	9.586	10.765	10.864
	10	3.156	3.675	4.379	5.086
Reduction	3	11.465	14.133	14.673	13.629
	5	7.006	8.173	9.901	10.271
	10	2.762	3.162	3.648	4.178
Sum Basic	3	9.198	10.555	10.154	8.991
	5	7.288	7.610	8.295	8.075
	10	2.213	2.689	3.242	3.756
Gold Standard	3	8.325	10.198	10.472	9.965

Table 5

Accuracy of the TF*IDF classifier on the original CNN articles. The TF*IDF threshold is varied from 1.0 to 12.0. Peak accuracies are indicated in bold text. The peak range is for thresholds between 3.5 and 6.0.

Trial	**TF*IDF Threshold Used in the Trial Classification**											
	1.0	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
1	0.37	0.42	0.46	**0.57**	0.56	0.56	0.56	0.55	0.55	0.56	0.53	0.50
2	0.40	0.41	0.44	0.57	0.56	0.55	0.57	0.57	**0.59**	0.58	0.58	0.48
3	0.45	0.45	0.58	0.59	0.59	**0.60**	**0.60**	**0.60**	0.59	0.57	0.58	0.48

Table 6

Accuracy of the classifier on the 3-sentence summaries, CNN articles. LexR=LexRank, Red=Reduction, SumB=Sum Basic, Gold=Gold Standard, and Rand= Random (Mean of 10 samples). Peak accuracies obtained are indicated by bold text.

Trial	**TF*IDF Threshold Used in the Trial Classification**											
LexR	**1.0**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
1	0.39	0.44	0.49	**0.50**	0.49	0.49	0.47	0.48	0.47	0.43	0.39	0.36
2	0.39	0.36	0.46	0.46	0.46	**0.48**	0.47	0.47	**0.48**	0.46	0.44	0.36
3	0.38	0.42	0.43	0.44	0.46	**0.47**	0.46	0.46	**0.47**	0.43	0.45	0.39
LSA	**1.0**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
1	0.41	0.46	0.50	**0.52**	**0.52**	**0.52**	0.50	0.49	0.49	0.48	0.48	0.35
2	0.40	0.46	0.48	0.49	0.49	**0.51**	0.48	0.47	0.46	0.47	0.48	0.37
3	0.38	0.38	0.47	0.50	0.50	0.49	0.50	0.49	**0.51**	0.49	0.48	0.38
Luhn	**1.0**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
1	0.43	0.44	0.50	**0.50**	**0.50**	0.48	0.46	0.45	0.45	0.46	0.43	0.38
2	0.41	0.46	0.49	0.48	**0.49**	**0.49**	0.46	0.47	0.46	0.47	0.46	0.37
3	0.40	0.47	0.47	0.50	0.48	**0.51**	0.48	0.50	0.49	0.46	0.47	0.39
Red	**1.0**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
1	0.45	0.48	0.50	0.50	0.49	0.48	**0.51**	0.49	0.49	0.47	0.46	0.36
2	0.47	0.47	0.48	**0.51**	**0.51**	0.49	**0.51**	0.50	0.49	0.49	0.47	0.42
3	0.43	0.41	0.49	0.50	0.50	0.51	0.51	**0.52**	**0.52**	0.50	0.49	0.40
SumB	**1.0**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
1	0.30	0.35	0.42	0.41	0.42	0.40	0.40	**0.44**	**0.44**	0.42	0.39	0.31
2	0.30	0.37	0.38	0.42	0.42	0.43	**0.44**	0.43	0.43	0.41	0.41	0.35
3	0.28	0.35	**0.40**	0.37	0.38	0.38	0.38	0.39	0.38	0.38	0.39	0.31
Gold	**1.0**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
1	0.37	0.46	0.47	**0.49**	**0.49**	**0.49**	**0.49**	0.47	0.45	0.45	0.43	0.34
2	0.41	0.38	0.46	0.46	0.47	0.46	0.45	0.46	**0.48**	0.45	0.46	0.35
3	0.38	0.44	0.48	0.48	**0.49**	**0.49**	0.46	0.47	0.47	0.46	0.45	0.37
Rand	**1.0**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
1	0.28	0.32	0.38	**0.39**	**0.39**	**0.39**	**0.39**	0.38	0.38	0.36	0.36	0.29
2	0.28	0.31	0.35	0.37	**0.38**	**0.38**	0.37	**0.38**	**0.38**	0.37	0.36	0.31
3	0.27	0.32	0.36	0.37	0.38	**0.39**	**0.39**	**0.39**	**0.39**	0.36	0.35	0.31

Table 7

Accuracy of the classifier on the 3-sentence summaries, CNN articles. LexR=LexRank, Red=Reduction, SumB=Sum Basic, Gold=Gold Standard, and Rand= Random (Mean of 10 samples). Peak accuracies obtained are indicated by bold text.

	TF*IDF Threshold Used in the Trial Classification											
Type	**1.0**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**5**	**5.5**	**6.0**	**7.0**	**8.0**	**12.0**
LexR	0.39	0.41	0.46	0.47	**0.48**	**0.48**	0.47	0.47	**0.48**	0.44	0.43	0.37
LSA	0.40	0.44	0.48	**0.50**	**0.50**	**0.50**	0.49	0.48	0.49	0.48	0.48	0.37
Luhn	0.41	0.46	**0.49**	**0.49**	**0.49**	**0.49**	0.47	0.47	0.46	0.46	0.45	0.38
Red	0.45	0.45	0.49	0.50	0.50	0.49	**0.51**	0.50	0.50	0.49	0.47	0.39
SumB	0.30	0.36	0.40	0.40	0.41	0.40	0.41	**0.42**	**0.42**	0.40	0.40	0.32
Gold	0.39	0.43	0.47	**0.48**	**0.48**	**0.48**	0.47	0.47	**0.48**	0.45	0.45	0.35
Rand	0.28	0.31	0.37	**0.38**	**0.38**	**0.38**	**0.38**	**0.38**	**0.38**	0.36	0.36	0.30

Most of the summaries were classified with higher accuracy than their random counterparts of the same size. Only Sum Basic summaries did not classify at a significantly higher accuracy than the random summaries (Table 6). The classifier, when applied to the Sumy summarizations, seems to work best when using a TF*IDF threshold of between 4 and 5. When the threshold is low, several terms that may not provide any classification power are included. If the threshold is too high, many useful terms may be excluded, as they will not have a large enough TF*IDF. When using a threshold of 12, only the terms that occur in only one class will be used for classification. For example, if a word occurs only in the "business" class during training, then a document containing that term during testing has a high chance of being classified as "business". However, there is a chance that a document in testing might not have any word that is above the threshold, making it unable to be classified.

Table 7 provides the mean outcome of the three trials in Table 6. The peak accuracies are (0.48, 0.50, 0.49, 0.51, and 0.42) for (LexRank, LSA, Luhn, Reduction, and Sum Basic) summarizations. All but the Sum Basic summarizations provide classification accuracy as good as the Gold Standard (0.48). The Random Summarization sets do not provide comparable accuracy (0.36).

The peak Error Scores from Table 4 (13.4, 10.0, 14.2, 14.7, 10.2, 10.5, and 0.0) for (LexRank, LSA, Luhn, Reduction, Sum Basic, Gold Standard, and Random) correlate highly with the corresponding peak accuracies (0.48, 0.50, 0.49, 0.51, 0.42, 0.48, and 0.36), with R^2=0.776.

4. Discussion and Conclusions

4.1. Functional Approaches

This paper introduced two novel research areas to the field of natural language processing in general, and extractive summarization in specific. The first is the use of query behavior to grade the efficacy of the summarization approach used. Similarity in the ranking of documents in response to queries was used to demonstrate the relative *functional equivalence* of the summarized document sets (S) to the original document sets (OD). The closer the behavior of the S and OD sets in response to the query terms, the more similarly the two corpora can be employed for relevant linguistic purposes. In order to illustrate this approach in a non-superficial fashion, two greatly different corpora (one a set of novels from Project Gutenberg, the other a set of gold standard-enriched CNN articles) were evaluated. For the Novels, the Luhn and LSA algorithms were quantitatively shown to be the two (of the five Sumy summarization approaches employed) that best match S and OD query behavior. For the CNN articles, the LexRank and Luhn algorithms were quantitatively shown to be the two (of the five Sumy summarization approaches employed) that best match S and OD query behavior. Based on this functional metric, the Luhn summarizer is generally employable for creating summaries that respond similarly to a query set (and thus functionally have similar search behavior).

The second functional approach to assessment of summarization was the use of classification. Here, the CNN article set was directly relevant, since the documents are assigned to 12 classes of content. One concern with summarization – and a potential advantage of extractive summarization in comparison to abstractive summarization – is that the S set provides the same relevant content as the OD set. If a summarized set is capable of being classified as well as the original set, then this functional objective can be, at least in some sense, considered met. The results show that several of the summarization approaches (LexRank, LSA, Luhn, and Reduction) resulted in similar classification results as the CNN Gold Standard sentences, with or without TF*IDF term filtering (Table 7). These results match those of the query similarity results, or Error Score, of Table 4. The Sum Basic summarization approach provided similar results for the query functionality analysis, but not for the classification functionality analysis. It is proposed that the combination of these two functional tests is warranted for automated assessment of extractive summarization efficacy.

4.2. Summarization of the Two Widely Different Corpora

The overall summarization results for the Novels are dependent on the number of sentences chosen for the summary. This can be illustrated using comparisons to random sentence-assignment summaries. For the 20-sentence summaries, the LSA approach had an error score 8.8 standard deviations less than the mean for random summaries, while the Luhn approach had an error score 11.5 standard deviations less than the mean for random summaries. For the 50-sentence summaries, LSA error scores were 9.0 standard deviations below those of random summaries, while Luhn

error scores were 10.3 standard deviations less. For 100-sentence summaries, LSA and Luhn were 5.7 and 5.3 standard deviations below the scores for random summaries. Clearly, if the number of sentences in the summary were further increased, the value of the particular summarization approach in comparison to simply a random collection of sentences would continue to decrease. The linear regression equation for Normalized Error Score (NSE) versus the number of sentences ($N_{Sentences}$) in the document summary for the Novels is given by Eq. 8, for which the y-intercept is $N_{Sentences} = 217$.

$$NSE_{Novels} = -0.0049(N_{Sentences})+1.0635 \quad (8)$$

Based on this finding, a summary of the novels comprising more than 200 documents would not provide an advantage over simple collecting random sentences from the novels. Therefore, the length of summarizations selected (20, 50, and 100 sentences) are suitable for comparative and functional analysis of the Summarization algorithms.

Similarly, for the CNN Articles, the mean Error Score varies with the length of the summarization. The mean Error Score is 550.2 for three sentences; 348.6 for five sentences; and 161.6 for ten sentences. The linear regression equation for Normalized Error Score (NSE) versus the number of sentences ($N_{Sentences}$) in the document summary for the CNN Articles is given by Eq. 9, for which the y-intercept is $N_{Sentences} = 13$.

$$NSE_{Articles} = -0.0938(N_{Sentences})+1.2146 \quad (9)$$

Clearly, evaluation of 3, 5, and 10 sentence summaries for the CNN Articles was justified (particularly since the author-provided gold standard is three sentences).

The summarizers evaluated perform differently based on the severity of the TF*IDF filtering, typically performing optimally with TF*IDF in the range of 3.0-6.0. Of all of the summarization approaches teste, Luhn is the one recommended overall, for at least four reasons: (1) it performs as well as the LSA and better than the other approaches for query behavior of the Novels; (2) it performs as well as any other algorithm for query behavior of the CNN Articles; (3) it provides classification accuracy as high as the other algorithms, including the Gold Standard; and (4) it provides consistent classification accuracy over a very wide range of TF*IDF values (Table 7), indicating it is a resilient approach.

One factor that likely contributed to the relatively higher Error Scores for the summarizers on the novels is that the parser in Sumy can get confused by certain phrases. For example, some sentences that end with the name of a person with a title are parsed incorrectly. For example, the phrase “The sun shone on Mr. Smith.” Could be incorrectly parsed as two sentences: “The sun shone on Mr.” and “Smith.”. In novels such as *Pride and Prejudice*, where major characters have names in this format, such as Mr. Darcy, the invalid sentence “Darcy.” could end up being chosen as an important sentence, as the word “Darcy” is a very frequent word in the novel and would have a large term frequency. The Reduction algorithm in Sumy often chose these single word non-sentences as part of the summary, resulting in its relatively poor performance. The summarizers Luhn and LSA avoid shorter sentences, and thus almost never chose these fragment sentences as part of the summary. This finding supports the fact that the Error Score in Tables 1-4 is correlated with better summarization behavior.

One large difference between fiction (Novels) and non-fiction (ANN Articles) documents is the presence of dialogue. This can cause issues parsing and can also result in non-traditional sentences. In dialogue, it is not uncommon to have a sentence consisting of only a name, either as an exclamation or used in an introduction. This can lead to similar fragment sentences as mentioned in the previous paragraph. Dialogue can also cause parsing problems, such as when a character asks a question. A parser may have trouble deciding where to split the phrase “’How is the weather?’ asked Smith.” The parser in Sumy often split phrases like these into two sentences, resulting in sentence fragments such as “asked Smith.” These fragments were also commonly selected by the Reduction summarizer.

4.3. Interpretations and Future Work

It is worth noting that the methodologies for query-based and classification-based evaluation of summarization described herein is readily applied to both extractive summarization, as shown, but also abstractive summarization if desired. The classification technique, relying on a form of TF*IDF or even sentence weighting, is directly applicable to abstractive methods. With some modifications accounting for sentence similarities, the query approach would also be applicable to abstractive summarization. Thus, this paper provides a general-purpose methodology

for the evaluation of the relative accuracy of summarization for any existing summarization approaches along with those to come. The application of the methodology to two widely different data sets show that it is a general-purpose approach; for example, it could be used effectively on content related to the social sciences or physical sciences, as well.

The results, unsurprisingly, show considerable differences consistent with the different nature of these two data sets. The LSA and Luhn summarization approaches were most effective on the database of fiction (Novels), while the LexRank and Luhn summarization approaches were, arguably, most effective on the database of articles. Overall, the Luhn approach was deemed the most generally relevant. Based on the findings for the two functional metrics – query behavior and classification accuracy – it is clear that these two metrics can be used together to perform a "preflight" analysis of which summarization algorithm to employ on a task. As demonstrated here, comparing the results to those of a random-generate summary is also recommended. In addition, selecting a corpus to perform this preflight testing on is also important. For example, the Error Scores for all five Sumy summarizers are much less as a percentage of the maximum Error Score for the CNN Articles than for the Novels. This may be due to greater repetition in the articles, with random sentences more likely to hit on the article-differentiating themes. That is, randomly selecting from a short article is more likely to get gist of it than randomly jumping around in a novel.

Follow-up work in the intersection of multi-corpora and functional metrics is likely to focus on simultaneous optimization of multiple assessment metrics [23, 24]. In this way, the relative impact of different sentence attributes (word scoring, word frequency, upper case, proper nouns, word co-occurrences, lexical similarity, etc.) on the utility of the summarization can be addressed. Elements of this were addressed by the use of TF*IDF and synonymy in this paper. A future area for research on functional summarization is the use of regularization during summarization, as outlined in Equations 1-2. This was not an option for the Sumy summarizers. Additional insight into the saliency of assessing summarization via classification behavior may be gained through the incorporation of additional classification approaches [25, 26, 27]. The intent of this paper was to provide a methodology for incorporating existing summarization frameworks into analysis, so the evaluation of additional summarization frameworks is potentially valuable future work. For example, newer and more powerful and sophisticated supervised machine learning/classification algorithms could readily be incorporated into future extensions of this research, including enhanced probabilistic neural networks, neural dynamic classification, dynamic ensemble learning, and finite element machines [28, 29, 30, 31]. The methodology described herein will allow comparative evaluation among these and other approaches. The fact that the known "poorest" summarizers, Random and SumBasic, performed the poorest on both functional metrics, is evidence that the functional approaches introduced herein work.

This paper makes the argument that the main purpose of the summarization is to be used properly. Two functional approaches to assessing the suitability of summarization were addressed: (1) using matching of query behavior to provide summarizations suitable for search; and (2) using classification of summarized documents to provide summarizations with content relevantly distinguishable with the desired partitioning of content (classes). These metrics showed that several summarization approaches resulted in functional summaries comparable to the Gold Standard summaries for the CNN articles.

However, a functional process such as those outlined here, may obscure some of the important details of how the process is actually working. Future work, therefore, may include focusing on understanding the nature of the correlation between query and classification in the original and summarized document sets. It may well be that some form of more advanced partitioning of the input, consistent with sub-classes, may further improve the output.

5. Acknowledgements

The authors gratefully acknowledge the State of Colorado (SB 18-086) and the National Science Foundation (NSF Award #1842577) for funding parts of this research. Further information on the Python sumy summarization approaches can be obtained on sumy 0.9.0, https://pypi.org/project/sumy/, accessed 12 March 2022.

References

[1] Gupta, S., and Gupta, S.K. (2019). Abstractive summarization: An overview of the state of the art. Expert Systems with Applications 121:49-65.

[2] Luhn, H.P. (1958). The automatic creation of literature abstracts. IBM Journal of Research and Development 2(2):159-165.
[3] Edmundson, H.P., and Wyllys, R.E. (1961). Automatic abstracting and indexing—survey and recommendations. Communications of the ACM 4(5):226-234.
[4] Edmundson, H.P. (1969). New methods in automatic extracting. Journal of the ACM (JACM) 16.2:264-285.
[5] Rush, J.E., Salvador, R., and Zamora, A. (1971). Automatic abstracting and indexing. II. Production of indicative abstracts by application of contextual inference and syntactic coherence criteria. Journal of the American Society for Information Science 22(4):260-274.
[6] Jones, K.S. (1993) What might be in a summary? Information retrieval 93:9-26.
[7] Kupiec, J., Pedersen, J., and Chen, F. (1995) A trainable document summarizer. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 68-73. ACM.
[8] Basiron, H., Jaya Kumar, Y., Ong, S.G., Ngo, H.C., and Suppiah, P.C. (2016). A review on automatic text summarization approaches. Journal of Computer Science 12:178-190.
[9] Wong K.F., Wu, M., and Li, W. (2008). Extractive summarization using supervised and semi-supervised learning. In Proceedings of the 22nd international conference on computational linguistics 22:985-992.
[10] Kaikhah, K. (2004). Automatic text summarization with neural networks. In 2nd International IEEE Conference on 'Intelligent Systems'. Proceedings 1:40-44.
[11] Svore, K., Vanderwende, L., and Burges, C. (2007). Enhancing single-document summarization by combining RankNet and third-party sources. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning 448-457.
[12] Lins, R.D., Oliveira, H., Cabral, L., Batista, J., Tenorio, B., Ferreira, R., Lima, R., de França Pereira e Silva, G., and Simske, S.J. (2019). The cnn-corpus: A large textual corpus for single-document extractive summarization. In Proceedings of the ACM Symposium on Document Engineering:1-10.
[13] Nallapati, R., Zhai, F. and Zhou, B. (2017). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Thirty-First AAAI Conference on Artificial Intelligence.
[14] M. F. Mridha, M.F., Lima, A.A., Nur, K., Das, S.C., Hasan, M., and Kabir, M.M. (2021). A survey of automatic text summarization: Progress, process and challenges. In IEEE Access 9:156043-156070.
[15] Verma, R., and Lee, D. (2017). Extractive summarization: Limits, compression, generalized model and heuristics. Computación y Sistemas, 21(4), pp.787-798.
[16] Simske, S.J., and Vans, M. (2021). Functional Applications of Text Analytics Systems. River Publishers.
[17] See, A., Liu, P.J., and Manning, C.D. (2017). Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368.
[18] Nallapati, R., Zhou, B., Gulcehre, C., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023.
[19] Rush, A.M., Chopra, S., and Weston, J.A. (2015). Neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685.
[20] Gerlach, M., and Font-Clos, F. (2020). A standardized Project Gutenberg corpus for statistical analysis of natural language and quantitative linguistics. Entropy 22(1):126.
[21] Stroube, B. (2003). Literary freedom: Project Gutenberg. XRDS: Crossroads, The ACM Magazine for Students 10(1):3-3.
[22] Vans, A.M., and Simske, S.J. (2017). Identifying top performing TF* IDF classifiers using the CNN corpus. In Archiving Conference 2017(1):105-115.
[23] Ferreira, R., de Souza Cabral, L., Lins, R.D., de França Pereira E Silva, G., Freitas, F., Cavalcanti, G.D., Lima, R., Simske, S., and Favaro. L. (2013). Assessing sentence scoring techniques for extractive text summarization. Expert systems with applications 40(14):5755-5764.
[24] Oliveira, H., Ferreira, R., Lima, R., Lins, R.D., Freitas, F., Riss, M., and Simske, S.J. (2016). Assessing shallow sentence scoring techniques and combinations for single and multi-document summarization. Expert Systems with Applications 65:68-86.
[25] Khan, A., Baharudin, B., Lee, L.H. and Khan, K., 2010. A review of machine learning algorithms for text-documents classification. Journal

of advances in information technology, 1(1), pp.4-20.
[26] Korde, V. and Mahender, C.N., 2012. Text classification and classifiers: A survey. International Journal of Artificial Intelligence & Applications, 3(2), p.85.
[27] Kim, D., Seo, D., Cho, S. and Kang, P., 2019. Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec. Information Sciences, 477, pp.15-29.
[28] Ahmadlou, M. and Adeli, H., 2010. Enhanced probabilistic neural network with local decision circles: A robust classifier. Integrated Computer-Aided Engineering, 17(3), pp.197-210.
[29] Rafiei, M.H. and Adeli, H., 2017. A new neural dynamic classification algorithm. IEEE transactions on neural networks and learning systems, 28(12), pp.3074-3083.
[30] Pereira, D.R., Piteri, M.A., Souza, A.N., Papa, J.P. and Adeli, H., 2020. FEMa: A finite element machine for fast learning. Neural Computing and Applications, 32(10), pp.6393-6404.
[31] Alam, K.M., Siddique, N. and Adeli, H., 2020. A dynamic ensemble learning algorithm for neural networks. Neural Computing and Applications, 32(12), pp.8675-8690.