TL;DR: Automatic Summarization With Textual Annotations

Julian S. Griggs

2015

Advised by Ananda Gunawardena

Submitted to: Princeton University

Department of Computer Science

This Thesis represents my own work in accordance with University Regulations

Date of Submission: April 30, 2015

Abstract

Automatic summarization is a powerful means for compressing large quantities of text into manageable chunks for human consumption. Despite the growth in blogs and other platforms that facilitate human interaction with text, there have been relatively few studies aimed at incorporating the auxiliary annotation data provided by these platforms into the summarization task. In this paper, I introduce a suite of summarization algorithms that utilize textual annotations (highlights and comments) to effectively summarize text. Specifically, two of the systems developed, ATMS-A (Hdp) and the Hotspot Summarization are shown to outperform all competitive baseline systems.

Acknowledgements

First off, I would like to thank my advisor Ananda Gunawardena for his advice and guidance over the course of this thesis. I would also like to thank Andrea LaPaugh for agreeing to read over this culmination of my year's toil. Most importantly, I would like to thank my mom and dad, who have, perhaps unknowingly, been my biggest inspirations over all of these years. I owe all that I am, and will ever become, to their uncompromising love and support.

To the Squad: From Roberts Stadium, to Club 93, to A38, we made it!

To Lauren Lazo: You are my best friend. Thank you for everything.

"Out of the night that covers me,
Black as the pit from pole to pole,
I thank whatever gods may be
For my unconquerable soul.
In the fell clutch of circumstance
I have not winced nor cried aloud.
Under the bludgeonings of chance
My head is bloody, but unbowed.
Beyond this place of wrath and tears
Looms but the Horror of the shade,
And yet the menace of the years
Finds and shall find me unafraid.
It matters not how strait the gate,
How charged with punishments the scroll,
I am the master of my fate,
I am the captain of my soul."

– William Ernest Henley, *Invictus*

Contents

List of Figures

List of Tables

1. Introduction

Information availability in today's world is vastly different from what it was in the world of 25 years ago. Relative to the luxuries of information access in the modern day, 1990 can be viewed as a sort of "dark age". At that time, information was primarily confined to physical books, "looking something up" meant taking a trip to the local library, and access to news came in the form of the daily newspaper. Fast forward to the present day and the differences are truly staggering. Today, people can download more books to their computers than would fit in a personal library, the answer to practically any question is available with a simple Google search, and news updates are immediately pushed to mobile devices. This modern age is one marked by the immediacy with which we have access to nearly unbounded sources of information, a luxury that is now difficult to image living without.

With the exponential increase in the data available on the World Wide Web, we now face an entirely different problem from that faced in the information dark ages: how do we avoid being inundated with information? When a Google search is able to return access to millions of potentially relevant sources, we face the dual challenge of determining the most relevant sources and then effectively digesting the abundance of information provided. Luckily, search companies are able to help us a lot with the first problem by performing relevancy related rankings. However, the second issue is still one without a clear-cut solution.

Automatic summarization is one mechanism aimed at countering information overload. These systems are designed to take text (single text, multiple related texts, email correspondences, transcripts of television broadcasts, etc.) as input and to produce a concise summary of the most important points as output. However, enabling computers to identify the most salient portions of a text is far from trivial with there being a large amount of on-going research geared towards discovering the best methods for identifying, synthesizing, and condensing information.

This study builds off of previous research in the field to enable an exploration of a specific summarization situation: summarizing a text that humans have annotated. The goals of this study

are to:

- Provide a comprehensive overview of the field of automatic summarization.
- Propose novel summarization algorithms that incorporate auxiliary data in the form of textual annotations.
- Compare the effectiveness of our proposed algorithms to each other and competitive baselines.

2. Background

In this section I begin by defining what a summary is and then move on to explain the various types of summaries. Readers who are familiar with the jargon used in the field may wish to skip ahead to Section 3.

2.1. What is a summary?

Prior to launching into a full discussion of summarization, it is important to explore what exactly constitutes a summary. Stated plainly, a summary is the product of condensing the information present in a source into a smaller chunk. However, differences in the level of condensation, information prioritization, and types of input sources lead to the generation of summaries that all provide something different. Newspaper headlines, literature abstracts, and snippets from a Google search all are examples of different summary variations. In order to capture the ways in which summaries vary, the field of automatic summarization uses the following categories:

1. Means of Creation
2. Number of Sources
3. Content
4. Scope

2.2. Means of Creation: Extractive vs. Abstractive

The literature defines two methods for summary generation: *extraction* and *abstraction*. Extractive summaries are those that are produced through a process where the text's most important sentences are concatenated together without altering the sentences in any way. In other words, this method

of summary generation works by simply "extracting" the most relevant sentences from a text. In contrast, abstractive summaries are those in which the important themes from a text are identified and then new sentences are generated based upon a deeper understanding of the material. In other words, abstractive summaries are those created using a more "abstract" understanding of the material to generate a new sentence representation of it.

These two methods are representative of the ways in which humans summarize written material. Extractive summaries are akin to a human highlighting the parts of a text they deem important while abstractive summaries are akin to a human synthesizing the information they read, and then writing out the key facts in their own words. While on the one hand, the latter method is better for human recall, it also requires much more effort. As such, people often make use of highlights as a type of summarization heuristic; one that seeks to approximate the efficacy of generating a summary by one's self.

The difficulty differential between these two approaches is greatly increased when the task is handed over to computers. Even with the current state of the art in artificial intelligence, computers are still not nearly advanced enough to support the ability to "reorganize, modify and merge information expressed in different sentences in the input." [28] Thus, in an analogous manner to the way people simplify the task of identifying important information, most research in the field deals with the simpler, extractive based methods.

2.3. Number of Sources: Single-Document vs. Multi-Document

Methods of automatic summarization also differ based upon the number of sources of information that need to be summarized across. There are two categories that the source material can fall into: *single-document* or *multi-document*. Early work in the field dealt primarily with single-document summarization; producing summaries solely by looking at information present in one article, chapter, etc. However, with the advent of the World Wide Web came a large level of redundancy that motivated work in multi-document analysis [28]. With information regarding single topics present amongst multiple sources, the ability to be able to effectively synthesize information across

all of them became highly desirable. Though this work will deal primarily with single-document summarization, I will employ some techniques from the multi-document sphere as will be described later.

2.4. Content: Indicative vs. Informative

Summaries can serve one of two purposes: to indicate what the text is about, or to act as a document replacement. Summaries made with the purpose of the former, are known as *indicative summaries*, while those adhering to the latter purpose are known as *informative summaries*. Indicative summaries enable the reader to determine the "about-ness" of a document and often involve deriving a set of key words/phrases that are indicative of the document [28]. Informative summaries, on the other hand, are more comprehensive in that they seek to not only signal what the document was about, but actually provide enough content for the end user to develop an understanding on par with what a full read of the actual text would provide. Newspaper headlines are an example of indicative summaries while the abstract of a journal article is an example of an informative summary.

2.5. Scope: Generic vs. Query-Focused

The intended scope of the summary is the last factor that determines the type of summary to be produced. Scope can fall into one of two categories: *generic* or *query-focused*. Generic summaries are those that make no assumptions about the material to be summarized or the user for which the summary is being produced [28]. In this way, they are the most "generic" and the methods for producing these summaries are immediately accessible to all genres. Query-focused summarization differs from generic summarization in that the goal is instead to craft a summary that is most relevant to a provided user query [28]. In this setting, the user query is of equal importance to the document itself, thus requiring a different set of methods that are able to take both of these factors into account.

3. Surveying the Field of Automatic Summarization

In this section I will survey the field of automatic summarization, highlighting the most important approaches and methodologies. I begin by exploring Luhn's seminal work as a means of conveying the intuition behind frequency-based approaches. I then move on to discuss the development of machine learning techniques that attempt to use various document level features for creating summaries. I then transition to a discussion of graph-based methods culminating with the development of the LexRank algorithm. Finally, I provide a brief discussion of some of the work that has been done involving lexical chains.

3.1. Identifying Important Sentences

Once more, a summary is the product of condensing the information present in a source into a smaller chunk. Despite the simple definition, a large challenge lurks: how do we condense the information? Solving this problem relies critically upon an identification of the most important concepts present in a text. When space is limited, in order to remain representative of the source text, a summary must prioritize the most important themes. Enabling a computer to perform this task in the same way in which a human might, relies upon developments in artificial intelligence that are currently not accessible. Therefore, we must settle for techniques that approximate, or perhaps even outperform, the human based methods.

At the most fundamental level, the computer must somehow be able to determine what the text is about before it can find the sentences best representing the main points. In his pioneer work, H.P. Luhn determined that the significance of a sentence is contingent upon the significance of the words it contains. He went further to suggest that word frequency provides an effective measurement of word significance, and thereby sentence significance [22]. This notion, though having some kinks that Luhn himself pointed out, became the foundation of extractive based approaches. A word's significance is equal to its probability of occurring, which is defined by:

$$significance(w) = p(w) = \frac{c(w)}{N} \tag{1}$$

where:

$p(w)$ = probability of a word, w, occurring

$c(w)$ = number of times a word, w, occurs in the input document

N = total number of words in the input

Because this metric normalizes the frequency based upon the size of the input, it can be used just as well for the single and multi-document cases. Given the probability distribution over words formed by calculating the probability for each word as in Equation 1, the likelihood of a summary can be computed based on a multinomial distribution as shown in Equation 2 [28]:

$$L[sum] = \frac{M!}{n_1! \ldots n_r!} p(w_1)^{n_1} \times \cdots \times p(w_r)^{n_r} \tag{2}$$

where:

n_i = number of times word w_i appears in the summary

M = number of words in the summary, $n_1 + \cdots + n_r = M$

r = number of unique words in the input

With this metric established, one could imagine many different mechanisms for extracting sentences as part of a formative summary. One such method would involve calculating the probability of each word and then for each sentence in the document averaging the probability of each word to form a single metric representing each sentence. After this, the sentence with the highest average score would be extracted and then this process would loop until reaching a summary of the desired length. See Appendix 1 for details. Other algorithmic variations could involve recalculating the probabilities upon the extraction of sentence and/or optimizing globally over the entire summary as opposed to the greedy approach outlined [28].

As Luhn himself notes, one glaring problem with the previously described system is that it does not account for the fact that in the English language, the most commonly occurring words are "function words". Function words are those that serve the grammatical purpose of tying ideas together rather than being substantive in their own right [22] and include: "the", "and", "of", "a",

"it", etc. Luhn solved this issue by manually creating a "stop word list" that is comprised of common function words that should not be considered when determining word probabilities [28].

Another problem Luhn noted is the artificially high frequency with which domain-specific words might appear (ex. the word "algorithm" in a computer science text). Luhn realized that words pivotal to the domain at large do not actually offer much information about the topic of the text and should be ignored in a similar fashion as the function words. Since these words cannot be predicted ahead of time as easily as function words, Luhn chose thresholds in which he found word probabilities to be indicative of subject matter [22]. Though empirically validated, these thresholds were more or less arbitrary, paving the way for more data driven approaches that build off of his seminal work.

3.2. TF*IDF Weighting

One data driven approach that utilizes the power of word frequency as an importance indicator, while simultaneously making up for the shortcomings previously described is the Term Frequency*Inverse Document Frequency(TF*IDF) metric. The TF*IDF metric is based on the idea that the most important words are those that occur frequently within the given document but also infrequently in other documents of the same genre. TF*IDF derives its power through use of a large background corpus of documents related in genre to the test document through which a baseline word frequency distribution is generated. Mathematically, the TF*IDF score is calculated by the following [28]:

$$\text{TF*IDF}_w = c(w) \times \log \frac{D}{d(w)} \tag{3}$$

where:

$c(w)$ = frequency of word w

$d(w)$ = number of documents in the background corpus that contain w

D = size of the background corpus

This metric has the appealing property that both function words and commonly occurring domain-specific words, which will likely appear in all documents of the background corpus, will have a

TF*IDF score of 0, thereby reducing their importance in a much less arbitrary manner than the "stop word" list method used by Luhn. An additional, crucial factor, that lends to the usefulness of the TF*IDF scores is that their calculation is fast and easy to compute [28].

3.3. Beyond Word Frequencies

In expansion of Luhn's approach, researchers such as H.P. Edmundson thought it useful to look at more features of a text than just word-frequency; specifically, he looked at cue words, title and heading words, and sentence location [7]. For Edmundson, cue words are those words in a sentence that belong to a pre-compiled list of cue-phrases, title/heading words are those that appear both in a sentence as well as in the document's title or headings, and sentence location refers to the position of the sentence in the document [28]. An interesting finding of this work was that when each feature was taken by itself, the worst performing feature was in fact the word-frequency feature originally put forth by Luhn [28].

One major caveat to this result is that Edmundson's work was geared towards a single domain, chemistry articles. Thus, it can easily be argued that the features he chose took advantage of a domain specific knowledge that probably is not directly applicable to other domains. This work, occurring long before the invention of modern machine learning algorithms, relied on assigning manually generated weights to these four features. However, its multi-feature approach can be seen as the foundation on which machine learning approaches to summarization are based.

3.4. Machine Learning for Extraction

Developments in machine learning have had a rather significant spill over into the realm of machine summarization. With a multitude of features including word frequency, sentence location, sentence length, and title composition being suggested for use in identifying salience, having a statistical means to determine the best combination of such features is incredibly valuable. This main drawback of the machine learning based approaches is their dependence upon having access to a set of labeled training data. Researchers have found various ways to try to ease the burden of this requirement, but attaining such a labeled set is still costly in the normal case.

3.4.1. Naive-Bayes Methods

In early work using machine learning for summarization, Kupiec et al. used a Naive-Bayes classifier to determine the optimal weighting of a set of features for the summarizing a corpus of technical articles from the Engineering Information Company. The features they were looking at included the following [18]:

- **Sentence length:** Is the sentence length greater than a specified threshold value?
- **Fixed-Phrase:** Does the sentence contain a predetermined cue phrase such as "in conclusion"?
- **Location in Paragraph:** Does the sentence appear at the beginning, middle, or end of a paragraph? (Only paragraphs that occur towards the beginning and end of the document are considered)
- **Thematic Words:** Does the sentence contain many frequently appearing content words?
- **Uppercase Words:** Does the sentence contain many capitalized words?

Their results indicated that a combination of location in paragraph, fixed-phrase, and sentence length yielded the best results with the incorporation of thematic words actually leading to poorer performance [18]. Kupiec et al. attribute this surprising result to the fact that their corpus contained texts that mostly had indicative material that was clumped around the tail ends of the document, leaving the middle relatively sparse [18]. Though a very promising result, their use of a Naive-Bayes classifier requires the assumption that features are independent of one another, an assumption that probably is not wholly true.

3.4.2. Hidden Markov Model

In order to try to relax some of the required independence assumptions required by application of a Naive-Bayes Classifier, Conroy and O'leary instead applied the Hidden Markov Model (HMM) to the summarization task (see [10] for more information about HMM). They predicted that the probability of one sentence being included in a summary is dependent upon whether or not the previous sentence was included. This hypothesis naturally motivates the use of an HMM, as the model does not require independence between sentence i and sentence $i-1$ [6]. Application of this method led to the generation of summaries that outperformed the Naive Bayes approach in terms of

their likeness to human expert generated summaries [6].

3.5. Graph-based methods

Graph-based summarization methods have been shown to be useful for both single document and multi-document summarization due to their ability to incorporate word frequency information into a formalized framework within which to analyze sentence-to-sentence relationships [28]. The foundational assumption made by graph-based methods is that the sentences with the greatest similarity to the other document sentences are the ones most salient to the topic [1]. In order to find these most central sentences, graph-based methods build a document graph where sentences are the vertices in the graph with edges connecting related sentences. In order to quantify the notion of "related sentences" a similarity metric is often used as the edge weight between two vertices with an occasional variant being to connect vertices in a binary fashion (connected only if similarity metric is above a given threshold).

The standard metric used for calculating the similarity between two sentences is to compute the cosine similarity between their TF*IDF weights [28]. In order to use this method, sentences are treated as N-dimensional vectors where N is the number of uniquely occurring words in the document. The vector values are first initialized to 0, and then for every word in the sentence, the corresponding element in the N-dimensional vector is set to that word's TF*IDF weight.

$$\mathbf{V}(s_i) = < f_{w_1}, f_{w_2}, \ldots, f_{w_n} > \tag{4}$$

where:

$$f_{w_j} = \begin{cases} \text{TF*IDF}(w_j), & \text{if } w_j \in s_i. \\ 0, & \text{otherwise.} \end{cases}$$

After deriving the TF*IDF vector representation for each sentence, the similarity between two sentences is simply equal to the cosine distance between the two vectors [1]. Mathematically, the

cosine similarity between two sentences, s_1 and s_2 is as follows:

$$\text{Cosine Similarity}(s_1, s_2) = \frac{\mathbf{V}(s_1) \cdot \mathbf{V}(s_2)}{||\mathbf{V}(s_1)||\ ||\mathbf{V}(s_2)||} \tag{5}$$

In this equation, the numerator is simply the dot product of the two vectors while the denominator is the product of their Euclidean lengths. The purpose of the denominator is to normalize the vectors $\mathbf{V}(s_1)$ and $\mathbf{V}(s_2)$ to unit vectors [23]. Using cosine distance as the inter-sentence similarity metric, a similarity graph can be generated through calculation of the similarity between every pair of input sentences and setting their edge weight to the calculated value. This graph representation can then be analyzed using various techniques.

3.5.1. Degree Centrality

One graph analytic technique that can be applied is degree centrality. The degree centrality of a sentence is defined as the in-degree of its corresponding node in the similarity graph. In explanation, consider Table 1 below which is a matrix representation of the cosine similarity between four sentences: *w*, *x*, *y*, and *z*. Sentences are intrinsically similar to themselves, hence the value 1 across the diagonal of the matrix.

	w	**x**	**y**	**z**
w	1	0	0.2	0.3
x	0	1	0.4	0.1
y	0.2	0.4	1	0.2
z	0.3	0.1	0.2	1

Table 1: Example Cosine Similarity Matrix

When building a similarity graph from the matrix representation, we first must decide upon a similarity threshold for adding edges to the graph. The motivation behind this is simple: we are only interested in looking at sentences that are similar in a significant way. If we do not set a threshold value, then we will be adding edges connecting loosely related sentences (the lower values in the matrix), which will render the degree centrality of a sentence meaningless. Choosing an appropriate threshold value is crucial for achieving a valid interpretation of centrality. As explicated by Erkan et al., “too low thresholds may mistakenly take weak similarities into consideration while too high

thresholds may lose many of the similarity relations." [1]. This exact observation is demonstrated in Table 2.

ID	Degree(t=0)	Degree(t=0.1)	Degree(t=0.2)	Degree(t=0.3)
w	4	3	3	2
x	4	3	2	2
y	4	4	4	2
z	4	4	3	2

Table 2: Degree Centrality for Table 1. Degree(t=x): degree of node when only counting edges with cosine similarity $\geq x$

When the threshold is set at $t = 0$, the degree of all nodes must be identical as 0 is the lower bound of the cosine similarity. Thus no useful information is gleaned. Similarly, when setting the threshold too high, such as $t = 0.3$, we have eliminated some of the useful, differentiating that is present at the $t = 0.2$ level. Thus, for this example, setting the threshold value at $t = 0.2$ would probably be appropriate.

3.5.2. LexRank

Erkan & Radev utilized the PageRank algorithm [2], as well as degree centrality, to analyze the similarity graph in a method they termed "LexRank". In their first approach to creating LexRank, Erkan & Radev opted to not use any type of edge weighting factor. Instead, their document graph included directed edges connecting sentences in a binary fashion; two sentences were connected only if their cosine similarity was greater than a given threshold value, a process identical to the thresholding discussed in Section 3.5.1. Since cosine similarity is a commutative metric, any edge added from $V_a \rightarrow V_b$ would have the same weight as the corresponding edge from $V_b \rightarrow V_a$. After generating the graph representation of a document, PageRank was applied to the graph, with sentences being ranked an extracted in order of their PageRank scores.

Erkan & Radev found that this method was able to extract the most important sentences of the document, in the best case, better than all other baselines of the time [1]. This method, though promising, suffers a similar problem as determining degree centrality: how does one set the arbitrary threshold values? In order to account for this, Erkan & Radev expanded LexRank into what they termed, Continuous LexRank. Rather than the binary thresholding performed on the cosine

similarity matrix performed in standard LexRank, the idea behind Continuous LexRank is instead to directly utilize the similarity metrics as an indicator of the "strength" of the links [1]. In other words, the idea is to incorporate the cosine similarity into weighted edges connecting two sentences in the graph.

After normalizing the row sums of the "weighted transition matrix", the resulting matrix is stochastic and therefore able to efficiently run the PageRank algorithm over. Though the resulting matrix is less sparse, and therefore takes longer to converge, Continuous LexRank gains the advantage of not requiring arbitrary threshold values for its application. Despite this benefit, their study found that when using what they determined to be an optimal threshold value of 0.1, degree centrality, LexRank with threshold, and Continuous LexRank all performed at about the same level of efficacy [1].

3.6. Lexical Chains

A large body of work in the summarization field deals with attempting to make sophisticated use of language semantics in order to identify important topics in a text. The use of lexical chains is one such method. Lexical chains can be defined as sequences of semantically related words ranging across an entire text [27]. In demonstration, consider the following piece of text:

"The soccer player kicked the ball and scored a goal."

In the example above, the words "soccer", "kicked", "ball", and "goal", form a lexical chain; when taken in aggregate, they indicate an obvious topic. Therein lies the intuitive power behind the use of lexical chains; by forming connections between the words in the text, the underlying topical structure of a text is uncovered. Approaches that make use of lexical chains rely heavily on WordNet [25] for its ability to provide information about word meaning and relationships between words.

In a formative study involving the use of lexical chains for summarization, Barzilay and Elhadad developed a summarization system that worked in four parts [3]:

1. **Segment the original text:** In order to segment the text, Barzilay and Elhadad used Hearst' TextTiling algorithm which uses lexical frequency and distribution to split a text into units reflecting the topic structure of a text [13].
2. **Construct lexical chains:** The construction process takes place in 3 steps: selecting the candidate words, finding the chain that each word most strongly fits into, and inserting the word into the selected chain. Barzilay and Elhadad's work expanded upon this step by trying to optimize the lexical chains by taking into account all possible meanings (sense) of their candidate words (nouns) [3].
3. **Rank lexical chains by their strength:** For ranking chains, they developed a strength metric that is a product of:
   - length = number of words in the chain
   - homogeneity index = $1 - \frac{\text{the number of distinct occurrences}}{\text{length}}$
4. **Extract the most significant sentences:** The best heuristic that Barzilay and Elhadad came up with was, for each strong chain, to extract the first sentence in the text that contains a representative word for the chain. Representative words are defined as having a frequency in the chain no less than the average word frequency in the chain [3].

4. Related Works

There has been relatively little research that attempts to incorporate human annotations into the document summarization task, especially when it comes to the types of complex annotations that platforms like Classroom Salon allow. Such platforms allow annotations that not only serve to mark important regions of the text (via tools like highlighting) but also allow the annotator to generate their own questions, comments, or discussion in the form of human language that is associated within a particular textual context.

One study by Zhang et al. deals with a more simplified type of annotation: highlighted regions. Based upon the realization that annotations reflect user preferences, they hypothesized that summaries generated using annotations would be better as they are more tailored to the user interests

[38].

In order to incorporate annotations into the summarization procedure, Zhang et al. define three non-mutually exclusive categories that each word in the text may fall into:

1. **Text keyword:** This category contains all words with a TF*IDF frequency satisfies a given threshold. They are words that appear frequently in the document.
2. **Annotated keyword:** This category contains all the key words that occur in annotated regions.
3. **Context keyword:** This category includes all keywords that occur in sentences that include an annotation.

The final measure of importance for each word is found by performing a linear combination across these three categories [38]. The following relation explains this mathematically:

$$\begin{aligned} f_i &= F0_i + \beta F1_i + \gamma F2_i \\ F &= < w_i, f_i >, 1 \leq i \leq n, n = size(F) \end{aligned} \tag{6}$$

where:

F = Set of final word weights

f_i = final word weight

$F0$ = text keywords

$F1$ = annotation keywords

$F2$ = context keywords

β = annotation weight ($\beta = 1$)

γ = context weight ($\gamma = 1$)

Finally, sentences are awarded a weight equal to either the sum of the weights of all the words in the sentence or the sum of all the word weights normalized by length of the sentence. Once each sentence is weighted, the sentences are ranked and the top ones are used to create the summary, maintaining their original position within the text.

After testing this method, they found their hypothesis to be correct with experiments demonstrating that annotation based summaries had an average of a 6.47% increase in the cosine similarity

with "gold-standard" summaries over the strictly word-frequency based variant ($\beta = 0, \gamma = 0$) [38].

Despite having demonstrated that using personalized annotations is able to improve the summarization task, Zhang et al. found that improvement was not linear with the number of annotations. Rather, they found that in almost every case, the best summaries were produced using $0 < k < N$ annotations where N is the total number of annotations made [38]. Another critical component of the study by Zhang et al., as it relates to this project, is in their exploration of "collaborative filtering". They wanted to determine if annotations made by one person would help create a better summary for a different individual. This is particularly relevant to my project because I am proposing to utilize all annotations of a document, as a whole, in crafting summaries.

Their results reveal that the usefulness of applying the annotations of one individual on the creation of a summary for another is dependent upon the two individuals in question; sometimes the impact is positive and sometimes it is negative. Individuals for whom the impact is positive can be said to have similar interests in the document to the original annotator and those who the annotations impact negatively can be said to have different interests [38].

It is important to note that this study only used annotations created by a total of five people, and as such was unable to have a corpus of annotations anywhere near equal to the one generated by Classroom Salon. Therefore, it would be unreasonable to immediately discount the usefulness of aggregate annotations on the summarization task. However this study does bring up the interesting idea of grouping individuals based off of annotation similarity as a means of providing the best possible summaries.

This idea of using annotations to produce more personalized summaries was explored and expanded upon by Moro et al. in its relationship to an educational system called Adaptive Learning Framework (ALEF). They not only included personal annotations, but also domain specific knowledge to craft better summaries. Specifically, they made use of "domain concept models" created by experts in ALEF. These "domain concept models" reflected the important concepts in the domain of the text in addition to the relationships between them [26]. They were then able to combine these "domain concept models" with ALEF data corresponding to each student's knowledge of the

concepts, as well as student annotations, to craft personalized summaries. They found their method to produce better summaries, as evaluated by users in 49% of cases while producing summaries on par with a non-personalized summary in 20% of cases [26]. This finding, while promising, relies on a system that maintains large levels of domain specific information. These systems are not widespread and as such, alternate means for improving summary performance must be explored.

4.1. Starting to Utilize Comments in Summarization

The growing popularity of social websites, particularly blogs, has motivated early research into the application of summarization to this new medium. Blogs differ from traditional forms of text in that they also come with comments generated by their readership. Researchers working to incorporate comment data into the summarization of blogs assume that comment data will provide additional insights into the salient features of the document. In justification of this assumption, Hu et al. report, "First, despite their informal tone and style of writing, comments represent readers' understanding or feedback about a Web document's content. By considering these comments, the generated summary can better capture the input from the readers, as opposed to the author of the document only." [15] Since generated summaries are likely to find the most use for the readers of a text, rather than the authors, it makes even more sense to generate summaries that account for the reader's understanding of the material.

Hu et al. put forth a few novel approaches for appropriately utilizing comments for the task of document summarization. One of their fundamental ideas is that the score awarded to every word in the document is a linear combination of two separate scores: a document score and a comments score. The document score is simply the TF*IDF value of the word [15], but the comments score, on the other hand, is novel. The first step to deriving a word's "comment score" is to build up three separate graphs connecting the collection of comments:

1. **Topic Graph:** The main idea behind this graph is that comments are topically related if they contain a similar set of words. Thus, comments are connected to each other via edges weighted (using cosine similarity, for example) to account for the level of similarity. This graph is

bidirectional as it assumes that the similarity metric used is bidirectional in nature.

2. **Quotation Graph:** This graph describes a different type of connection between comments; any comment that directly quotes material from another is connected on the level of quotation. This graph is formed by adding edges in a binary fashion: if comment_a directly quotes material from comment_b then an unweighted directed edge is added from the node representing comment_a to the node representing comment_b.
3. **Mention Graph:** Similar to the quotation graph, this graph seeks to link sentences that reference each other. This graph differs from the quotation graph in that rather than forming an edge when a direct quotation is involved, an edge is only formed when one comment more generally refers to another, oftentimes as evident through a reference to a comment's author.

Once each of these graphs is formed, they are merged into a single multi-relational graph, on which the PageRank algorithm is used to determine the most important comments in the collection. Upon creating this hierarchy, each word of the document attains its "comment score" by aggregating the PageRank score of each comment that the word can be found in. Once each word of the document is scored, the sentences with the highest average score are extracted for summarization.

5. Evaluation Metrics

Arguably the biggest challenge being faced by researchers in the summarization space is finding adequate means of assessing the performance of their summarization systems. Summaries are meant for human use, and therefore their quality is inherently subjective, a fact that is made vastly more difficult to cope with given the widespread disagreement between individuals about what a "good" summary even is. In demonstration of this point, a study by Rath et al. revealed that manually generated summaries created by six different human judges for 10 Scientific American articles only had an average of 8% overlap between them [32]. In order to mitigate this difficulty, researchers have developed systems that either limit the need for subjective human involvement or remove the need altogether.

5.1. Recall and Precision

Most summarization evaluation systems require the judgment of a human, or set of humans, at some level. The most common requirement of these systems is a human produced "Gold-Standard" summary that acts as a model for which automatically generated summaries can be compared to. The two most commonly used evaluation metrics that relate an automatically generated summary to the gold-standard are "recall" and "precision". As defined by Nenkova and McKeown, "Recall is the fraction of sentences chosen by the person that are also correctly identified by the system and precision is the fraction of system sentences that were correct." [28] Put slightly differently, recall measures how well a summarization system limits false-negatives whereas precision measure how a system limits false-positives. When a gold-standard summary contains the same number of sentences system-summary, precision and recall are equal. However, when summary lengths are non-binding, *recall* is usually the preferred metric. The reason behind this preference relates back to the disagreement amongst human summarizers; a sentence deemed unimportant by one might be critical to another. This tendency to disagree means that the precision metric is unfair because it punishes systems for including sentences not found in the gold-standard, when such a sentence may very well have been included by a different gold-standard summary.

5.2. *ROUGE*

The Recall-Oriented Understudy for Gisting Evaluation (*ROUGE*) is a set of evaluation procedures that are able to automatically determine the quality of a system generated summary by comparing it to gold-standard summaries through use of n-gram, word sequences, and word pair overlaps [37]. *ROUGE* was developed to augment, and eventually replace, manual evaluation methods that required humans to determine the similarity between system and human generated summaries. There were two primary reasons for this:

1. During the human task of determining system and model overlap, human judges disagreed with their own prior judgments in up to 18% of cases [20].

2. The manual labor required was both time and cost prohibitive for the development of novel summarization methods.

The *ROUGE* system includes multiple variants including: *ROUGE*-N (n-gram recall), *ROUGE*-L (Longest common subsequence), *ROUGE*-W (Weighted longest common subsequence), and *ROUGE*-S (Skip-Bigram Co-Occurrence Statistics).

ROUGE-N measures the "n-gram recall between a candidate summary a set of reference summaries." and is calculated as follows [37]:

$$\textit{ROUGE}\text{-N} = \frac{\sum_{S \in \text{Reference Summaries}} \left(\sum_{\text{n-gram}_m \in S} \textit{Count}_{match}(\text{n-gram}_m) \right)}{\sum_{S \in \text{Reference Summaries}} \left(\sum_{\text{n-gram}_m \in S} \textit{Count}(\text{n-gram}_m) \right)} \tag{7}$$

where:

$\textit{Count}_{match}(\text{n-gram}_m) =$ the maximum number of co-occurring n-grams

ROUGE-N has a couple nice properties. For one, it is a recall measure, as made evident by the denominator of Equation 7 which sums the number of n-grams found across the reference summaries as opposed to the candidate summary. In addition, *ROUGE*-N assigns a heavier weight to n-grams in the system-summary that are present in multiple reference summaries. This is a good property because of the intuition that the best system summaries are those that capture the consensus amongst the various reference summaries [37].

ROUGE-L measures the longest common subsequence (LCS) between a system-summary and a gold-standard with the intuition being that the longer the LCS, the more similar they are. The biggest differentiating factor between *ROUGE*-N and *ROUGE*-L is that while the former only accounts for adjacent words, while the latter is able to capture all of the in-sequence word matches. In explanation, consider Table 3. S1 has a *ROUGE*-2 score of 2 because it matches the 2-grams "birds were" and "were chirping" while having a *ROUGE*-L score of 3 due to the match "birds were chirping". S2 also has a *ROUGE*-2 score but due to slightly different matches: "The birds"

and “were chirping”. Unlike S1, S2 has a *ROUGE*-L score of 4 due to it containing the words “The”, “birds”, “were”, and “blue” all in the same sequence order as they are found in the reference sentence.

Reference Sentence: “The birds were chirping.”

ID	Text	*ROUGE*-2	*ROUGE*-L
S1	"The birds, which were bright blue, were chirping."	2	3
S2	"Everywhere, birds were chirping."	2	4
S3	"The birds were chirping."	3	4

Table 3: My caption

One interesting consequence of the *ROUGE*-L method is that S2 and S3 have the same *ROUGE*-L score despite S3 being an exact copy of the reference sentence while S2 is not. In situations like these, *ROUGE*-L cannot properly rank the more precise S3 over S2. *ROUGE*-W accounts for this situation through a weighting scheme that assigns higher weights to consecutive words than to words which, though occurring in-sequence, are separated by other text.

The final method, *ROUGE*-S is very similar to *ROUGE*-2 with the notable exception that rather than only counting the number of consecutive 2-grams, *ROUGE*-S measures the number of overlapping skip-bigrams where skip-bigrams are defined as any in-sequence 2-word pairing [37].

All of the different *ROUGE* variations achieved high levels of correlation with human calculated overlap, especially when gold-standard evaluations were available for each document. Specifically as it relates to single document summarization, *ROUGE*-2, *ROUGE*-L, *ROUGE*-W, and *ROUGE*-S all achieved correlations with manual evaluation of up to .99 [37]. Its effectiveness and relatively low costs have made *ROUGE* the de facto for summary evaluation.

5.3. Incorporating Human Judgment

As it currently stands, the most popular manual method for summary evaluation is the Pyramid approach. Developed by Nenkova & Passonneau, the Pyramid method is a process in which human annotators perform fine-grained inspection of multiple human crafted summaries in order to determine which Summarization Content Units (SCUs) are the most important. SCUs are

not rigorously defined items, but can be thought of as clauses that resemble particular factoids [29]. Consider the following two sentences that come from different manually drafted summaries (example borrowed from Nenkova & Passonneau [29]):

1. In 1998 two Libyans indicted in 1991 for the Lockerbie bombing were still in Libya.
2. Two Libyans were indicted in 1991 for blowing up a Pan Am jumbo jet over Lockerbie, Scotland in 1988.

The first step in identifying an SCU is to group similar sentences together. Once grouped, related subcomponents become SCUs. In the example above, the underlined portions form one particular SCU, with others being present as well. After SCUs are located, they are assigned weights according to the number of summaries they appear in. Once weighted, the SCUs are organized into a pyramid hierarchy, with the SCUs occurring in the most summaries appearing at the top and the SCUs appearing in the least number of summaries at the bottom. Once this pyramid structure is determined, summaries are given a score equal to, "a ratio of the sum of the weights of its SCUs to the sum of the weights of an optimal summary with the same number of SCUs. It ranges from 0 to 1, with higher scores indicating that relatively more of the content is as highly weighted as possible." [29] By making use of multiple human generated summaries, and through careful extraction of their important content, the pyramid method is able to alleviate the problem of human summary variance.

5.4. Moving Away from Human Judgment

Despite the massive usability improvements offered by *ROUGE*, one large hindrance remains: gold-standards are still required. An accurate evaluation system that is not dependent upon access to gold-standard summaries would increase research opportunities in the field of automatic summarization by removing the gold-standard acquisition barrier. Work by Louis & Nenkova [21] is some of the first that begins to make this type of human-free evaluation possible. They propose a method called *SysSumm* that works by taking multiple machine-generated summaries (generated through different algorithms) and using their collective knowledge as a machine generated gold-standard.

The first step in *SysSumm* is to generate a pooled word distribution over a set of summaries

produced by high-performing baseline systems. In explanation of the motivation behind this step, Louis & Nenkova reason, "In this distribution, the content selected by multiple systems will be more prominent, representing the more important information." [21] With this gold-standard distribution in hand, *SysSumm* works by comparing it to the word distribution of a candidate summary. Since the gold-standard distribution is presumed to be an accurate representation the relative importance of the text's content, candidate summaries with similar distributions are assumed to be better than those with differing distributions. In order to actually measure the similarity between the two distributions, *SysSumm* uses Jenson-Shannon (JS) Divergence.

The major assumption made by *SysSumm* is that the baseline systems used to generate the gold-standard distribution are high performing. Louis & Nenkova demonstrate that by collecting the output of seven algorithms (Lead-Sentence, Mead, SumBasic, Topic Signatures, Graph-Based Method, LSA, and Greedy KLSum), *SysSumm* achieved a correlation of 0.91 with Pyramid evaluation, lending credibility to its use as a viable method for summary evaluation [21].

The seven algorithms will now be described:

- **Lead-Sentence Baseline:** This method is most often used in multi-document summarization. A summary is generated by taking the first sentence from each document, then the second, etc. until a document of the desired length is achieved [21]. As it applies to single document summarization, the natural corollary is to apply the same procedure to each paragraph unit in the text.
- **Mead:** This method chooses sentences for a summary via a weighted combination of three factors: centroid value (how central to the document theme the sentence is), positional value (where in the document the sentence appears), and first-sentence overlap (the degree to which the sentence is similar to the first sentence in the document) [31].
- **SumBasic:** SumBasic is a frequency based summarization system. The general idea is that sentences that contain the most commonly appearing content words are the most important. More specifically, this method calculates the probability of each content word occurring in the input, calculates the average probability across all words in a sentence, and then extracts the sentences with the highest probability. The novel aspect of this summarization scheme is that after

extracting a sentence, a reduction factor is applied to the probability of all the words appearing in the sentence so as to minimize redundancy in the summary [30].

- **Topic Signatures:** This method utilizes the extraction of topic signatures (sets of related words) in order to rank sentences according to how well their content words fit into the topic signatures [19]. This method is most often applied to the task of multi-document summarization due to its reliance on both a relevant, and a non-relevant set of documents.
- **Graph Based Method:** There are a variety of graph based methods: notably LexRank. LexRank was described in Section 3.5.2.
- **LSA:** This method will be discussed in in a Section 6.1.1
- **Greedy KLSum:** KLSum is an algorithm that picks sentences for extraction through an attempt to minimize the Kullback-Leibler (KL) divergence between the probability distribution over words for the background corpus and for the chosen summary. Due to the large computational complexity of finding the global minimization of this metric (all combinations of sentences must be checked), a greedy approximation is to iteratively add the next document sentence that yields the minimum KL divergence [12].

6. Incorporating Annotations Into Summarization

In this section I propose multiple algorithms for incorporating human annotations, which contain highlighted regions and comments, into the summarization task. I begin by introducing topic modeling, with a brief description of the three primary algorithms used in this space. I then propose three algorithmic variations of an approach I term Aggregate Topic Model Similarity (ATMS) that combine the use of topic modeling with the comments component of the user annotations to summarize texts. Finally, I propose two algorithms for automatic summarization that make use of the highlighted regions made available by user annotations.

6.1. Topic Modeling

As explained by David M. Blei of Princeton University, "Topic models are algorithms for discovering the main themes that pervade a large and otherwise unstructured collection of documents. Topic

models can organize the collection according to the discovered themes." [5] The value in this approach is that it allows documents to be grouped together by the themes that run through them, a task that would otherwise require mass amounts of human annotation. There are three main algorithms used in this realm: Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), and Hierarchical Dirichlet Processes (HDP).

6.1.1. LSA

As explained by Nenkova and McKeown, "Latent semantic analysis (LSA) is a robust unsupervised technique for deriving an implicit representation of text semantics based on the observed co-occurrence of words." [28] In other words, LSA groups contextually related words into sets, and then provides a means for analyzing the similarity between these sets in addition to the words within each set. The first step in LSA is to take the input text and convert it into a matrix, $M_{i,j}$, where each row, i, stands for a unique word in the text and each column, j, represents a textual unit (most often sentences). The cells, c_{ij},of the matrix reflect the weight (TF*IDF) of word i in textual unit j.

Once this matrix, M, is created, it is broken down into a product of 3 component matrices using Singular Value Decomposition (SVD). Mathematically, this relationship becomes $M = U \cdot \Sigma \cdot V^T$ [28]. With reference to this step, Landauer et al. explain, "One component matrix [U] describes the original row entities as vectors of derived orthogonal factor values, another [V] describes the original column entities in the same way, and the third [Σ] is a diagonal matrix containing scaling values such that when the three components are matrix-multiplied, the original matrix is reconstructed." [36]

The next step in LSA is dimension reduction in which all but the first x columns from U, Σ, and V are set to 0. The final step is matrix reconstruction which creates a matrix M' using the lowered dimensionality of the component parts. This reconstruction step sets each cell value in M' as a linear combination of the retained dimensions of the three component matrices [36]. Together, dimension reduction and matrix reconstruction allow LSA to essentially "infer" how often a word is likely to occur in a textual unit based upon the other words present. For more specifics on LSA, see [36].

Gong & Liu were among the first to propose using LSA directly for summarization. Their method

utilizes the fact that every row, i, in the matrix V^T produced by SVD can be understood as a topic, with each column, j, reflecting a different textual unit. Furthermore, each cell is a measure of how well a particular $sentence_j$ reflects $topic_i$. With this understanding, their algorithm simply traverses over each row in V^T and selects the sentence with the highest value, continuing to do so until a summary of the desired length is achieved [11].

Jezek & Steinberger point out that the main issue with Gong and Liu's approach is that every topic is treated as being equally important to all others, an assumption they claim to be unfounded. In order to fix this, they suggest choosing the top k, sentences with the greatest combined weight across all topics, a method that would permit the possibility of choosing more than one sentences per topic [16]. Mathematically, this comes to computing:

$$\text{summary sentences} = \max_k ||\text{sentence vector}_i|| \ \ \forall \ \ \text{sentence vectors}_i \in \Sigma^2 \cdot V^T \tag{8}$$

6.1.2. LDA

The foundational idea behind LDA is that all documents in a collection are made up of multiple topics, with the proportion of each topic differing amongst documents. In this context, "topics" are defined as distributions over a fixed vocabulary of words [5]. LDA assumes that documents are approximately generated in the following fashion:

1. Assume there is a finite collection of topics that all documents are based upon
2. Choose a random distribution over the collection of all topics by sampling from a *dirichlet* distribution.
3. Choose a random topic from the topic distribution generated above
4. Choose a random word from the topic's distribution over its vocabulary

By assuming this process, the genius of LDA is to take a document, and using a probabilistic method run the above process in reverse to generate the original topics in addition to each document's distribution over them. The specifics of this algorithm can be found in [5].

6.1.3. HDP

HDP can be best thought of as an LDA extension that does not require the initial assumption of

a finite number of topics. Basically, the process is the same as LDA except that the number of topics is itself sampled from a *dirichlet* distribution [14]. Theoretically, HDP improves upon LDA in situations where there can be no reasonable knowledge about the number of topics in the text, since the HDP algorithm is one that essentially "learns" this value. Specifics of this algorithm can be found in [35].

6.2. An expansion to Applying Topic Modeling to Summarization

This study utilizes the three topic-modeling algorithms discussed previously (LSA, LDA, HDP), to incorporate the human comments provided by Classroom Salon into automatic summarization. I propose an algorithmic suite, generally referred to as Aggregate Topic Model Similarity (ATMS), which is built upon the belief that the comments contained in the Classroom Salon annotations represent important themes in the text. As such, this method is one that creates a summary by choosing the sentences that reflect topics most similar to those found in the comments, on an aggregate scale. Specifically, I propose three separate algorithms: ATMS-A, ATMS-S, and ATMS-Top.

6.2.1. ATMS-A

Aggregate Topic Model Similarity - Annotations (ATMS-A), is an algorithm that builds a summary by extracting document sentences that have the most similar topic distribution to the topic distribution derived from the full corpus of comments. The underlying assumption made by this approach is that the full corpus of user comments might contain a topic distribution that more closely reflects the main themes of the document than a topic distribution built upon the text itself. This approach is motivated by the assumption that user comments are the output of a mental process of information synthesis, one that utilizes human intelligence to factor out the less important details of a text. The core of the algorithm is as follows:

1. Begin with the full collection of all Classroom Salon user comments on the given document.
2. Convert this collection of comments into a matrix representation, M_{ij},where each row, i, represents a unique word in the body of comments, each column, j, represents a particular comment,

and each cell, $c_{i,j}$ represents a frequency metric for $word_i$ in $comment_j$.

3. Apply any choice topic modeling algorithm to M_{ij} to achieve an output topic space which is simply the projection of M_{ij} into a space with lower dimensionality (reflecting the topics).
4. For every sentence in the actual document, convert it into the same topic space as derived above.
5. Compute the cosine similarity between the topic distribution of the sentence and the topic distribution of each comment.
6. Add the sentences from the original document with the highest aggregate similarity score to the summary.

See Appendix 2 for pseudo code.

6.2.2. ATMS-S

Aggregate Topic Model Similarity - Sentences (ATMS-S), is very similar in nature to ATMS-A with the fundamental difference being that the initial topic space that is produced is based upon the document itself as opposed to the comments. Once this topic distribution is discovered, summary sentences are extracted according to which has the greatest similarity to the body of comments. The reason for generating a topic space based upon the original text as opposed to the comments is motivated by a rationale opposite to that presented in ATMS-A. Rather than assuming that the topics factored out by comments lack any useful topic information, it assumes that all topics present in a text are important and that through comments, humans tend to affirm the most important ones to the highest degree while still affirming lesser ones as well.

The core of the algorithm is as follows:

1. Begin with the document for which a summary will be generated.
2. Convert the document into a matrix representation, M_{ij},where each row, i, represents a unique word in the original text, each column, j, represents a particular sentence, and each cell, $c_{i,j}$ represents a frequency metric for $word_i$ in $sentence_j$.
3. Apply any choice topic modeling algorithm to M_{ij} to achieve an output topic space which is simply the projection of M_{ij} into a space with lower dimensionality (reflecting the topics).
4. For every sentence in the actual document, convert it into the same topic space as derived above.

5. Compute the cosine similarity between the topic distribution of the comment and each topic distribution of each sentence.
6. Add the sentences from the original document with the highest aggregate similarity score to the summary.

See Appendix 3 for pseudo code.

6.2.3. ATMS-Top

Aggregate Topic Model Similarity - Top (ATMS-Top), is essentially ATMS-S with an additional preprocessing step that reduces the number of comments to be used. Rather than using all of the comments, ATMS-Top only uses most representative comments, with representative comments being those that are most similar to the topic distribution across the entire body of comments. The motivation behind this comment filtering procedure is to remove noisy comments that either do not contribute valuable information (gibberish comments) or are comments that do not accurately reflect the general consensus. The algorithm for preprocessing is as follows:

1. Begin with the full collection of all Classroom Salon user comments on the given document.
2. Convert this collection of comments into a matrix representation, M_{ij},where each row, i, represents a unique word in the body of comments, each column, j, represents a particular comment, and each cell, $c_{i,j}$ represents a frequency metric for $word_i$ in $comment_j$.
3. Apply any choice topic modeling algorithm to M_{ij} to achieve an output topic space which is simply the projection of M_{ij} into a space with lower dimensionality (reflecting the topics).
4. Convert each individual comment into the same topic space as derived above.
5. Compute the cosine similarity between the topic distributions of comments for every comment pair.
6. Add the k comments with the highest aggregate similarity score to the summary.

See Appendix 4 for pseudo code.

6.3. Hotspots

The Classroom Salon platform is entirely based upon the notion that the collective knowledge of the users will provide greater insights into the textual material. This collective knowledge is manifested most obviously in the form of textual "hotspots". These hotspots are heat-mapped regions that express the most highly annotated segments of a text. As it applies to summarization, these hotspots are prime candidates from which to build a powerful extractive summary; after all, they are the sentences that actual readers have determined are worthy of attention. The presence of hotspots in a text motivated the development of the Hotspot Summarization algorithm. The algorithm itself is very simple once the required data is gathered; it builds a summary through the extraction of the n most highly annotated sentences in the document where n is equal to the desired length of the summary. The presence of these textual highlights also lends itself very well to the application of the summarization algorithm described by Zhang et al.

6.4. Sentiment-Based Hotspots

The Hotspot Summarization algorithm described above is perhaps overly simple in that it fails to consider the "why" behind an annotation. Specifically, the Hotspot Summarization algorithm only takes into account the number of times a sentence gets annotated, assuming that this number directly reveals the sentence's importance. Conceptually, this approach is flawed because not all annotations reflect positively upon the sentence(s) found in their document context. For example, annotations such as, "it is unclear how this section relates to the topic" or "this is confusing", signal unimportant and/or convoluted segments of the source text that probably should not be incorporated into a summary. The problem with the simple Hotspot Summarizer is that it treats negative comments like those in the previous examples in the same way it treats positive comments.

In order to begin to address this issue, I developed a Sentiment-Based Hotspot Summarizer. This algorithm is very similar to the simple Hotspot Summarizer except that rather than a sentence receiving a score equal to the number of annotations that reference it, a sentence receives a score equal to the aggregate positivity rating of the annotations that reference it. The fundamental

assumption made by this method is that textual regions that contain annotations with positive sentiments are more indicative of good summary candidates than textual regions marked by negative sentiment.

In order to determine the positivity of an annotation, I utilize the open source project SentiWordNet [8] which is an open source WordNet extension that is capable of providing "positivity" and "negativity" scores for all words in the WordNet dictionary. With this faculty in place, determining the average positivity of a sentence is simple: it is the average difference between each sentence word's "positivity" and "negativity" score as provided by SentiWordNet. Mathematically, this value can be computed as follows:

$$\text{Average Positivity}(A) = \sum_{i=1}^{n} \frac{Positivity(A_i) - Negativity(A_i)}{n} \tag{9}$$

where:

n = number of non-stop words in the annotation

A = an annotation made up of n words, $\{w_1, w_2, \ldots, w_n\}$

7. Implementation Details

In order to perform summarization, I extended upon the open source python summarization module, *Sumy* [4]. *Sumy* came with the built in ability to take a piece of text, parse it into paragraphs and sentences, and then create summaries using Luhn's algorithm [22], Edmundson's algorithm [7], LexRank [1], LSA [11], and TextRank [24]. In order to round out this already powerful suite, I added implementations of a Lead-Sentence Summarization, a Greedy KL Summarization algorithm [12] and SumBasic [30], all of which are considered strong baselines for comparison. Since *Sumy* was a module built solely for the standard text summarization task (i.e. no additional data other than the document), I had to extend *Sumy* to be able to receive data in the form of annotations as well. With this extension in place, I was able to implement Hotspot Summarization and Zhang's algorithm [38], both of which rely solely on the location of the annotation (i.e. the highlight) as opposed to

annotations actual textual content. I also implemented Sentiment Based Hotspot Summarization with the help of the previously described SentiWordNet extension. As ATMS-Top, ATMS-S, and ATMS-A all rely on topic modeling algorithms for their implementation, I made use of a third party Python package called *Gemsin* [33]. Gemsin comes with the capability to convert vectorized input into the topic space and then to perform similarity queries on the output. For a demonstration of *gemsin*, see https://github.com/JulianGriggs/sumy/blob/dev/gensim_demo.py. For all methods I followed standard procedure for removing "stop words" and stemming words appropriately, using the Natural Language Toolkit [17].

8. Methodology

I begin this section by describing the web-based learning platform, Classroom Salon. I then continue by presenting the Classroom Salon dataset used in this experiment. Next, I explain why I opted to use certain evaluation methods. I conclude this section by detailing the experimental setup required for the summary evaluation methods I chose to use.

8.1. Classroom Salon

Classroom Salon is a web-based learning platform that promotes collaborative engagement with study material in the form of video lectures and documents through its rich annotation and discussion capabilities. Classroom Salon is very easy to use; while reading through a text, a student will use their cursor to highlight a segment of text and then provide a comment expressing any thoughts or questions they have relating to that segment. Figure 1 demonstrates this process. One of the most powerful features of Salon is that when many people annotate the same document, the document's most heavily annotated regions, termed "hotspots", become visible as demonstrated by Figure 2. Since hotspots are the locations that received the most attention from readers, they intuitively serve as indicators for the document's most critical regions. This intuition is what underlies the Hotspot Summarization algorithm. An analysis of the dataset reveals that on average, a student will highlight 5.06 ($\sigma = 1.53$) sentences, or about 4.1% ($\sigma = 2.3\%$) of the total number of document sentences, for each document they read.

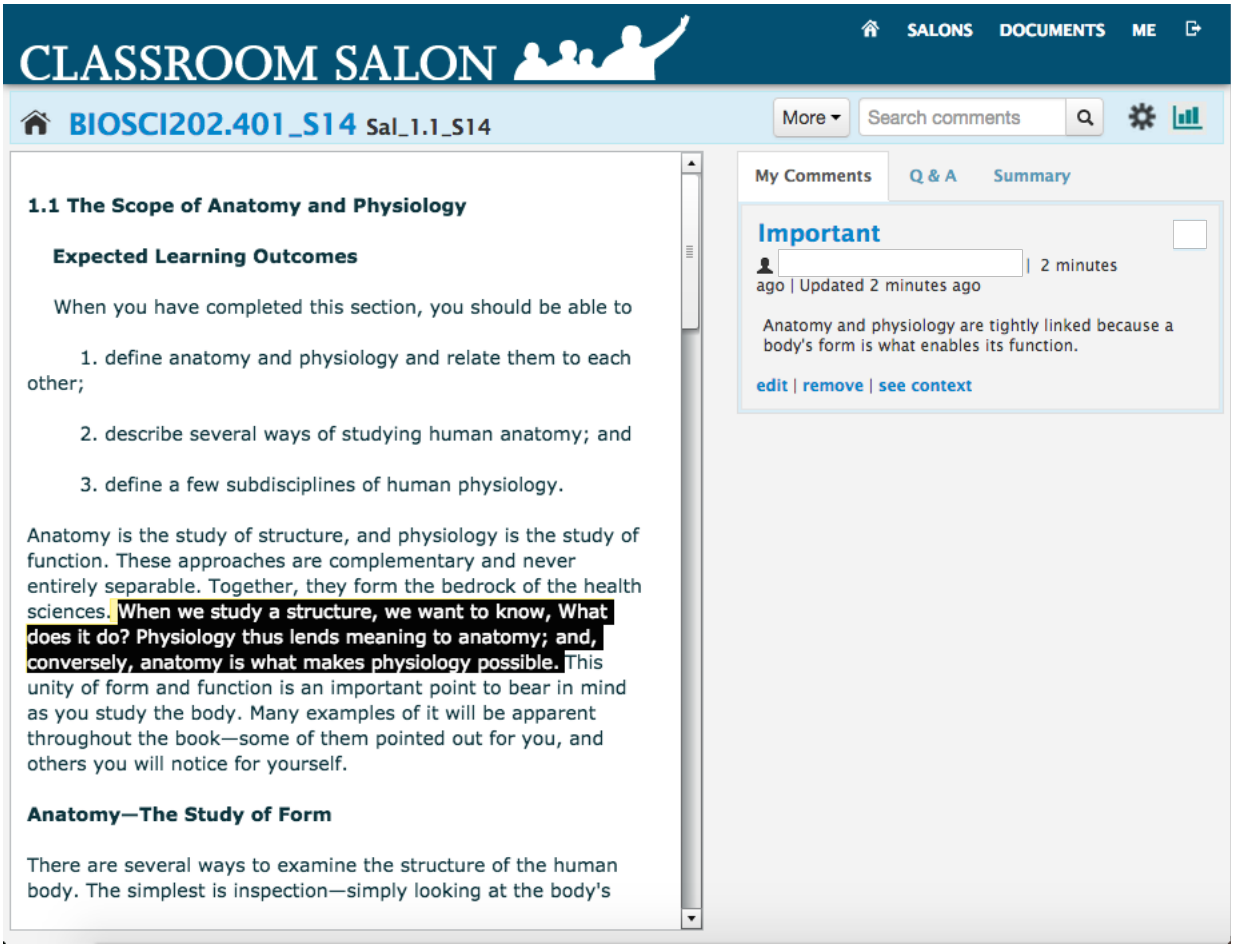

Figure 1: Sample Classroom Salon annotation. The comment in the right side of the image is linked to the highlighted region in the actual text through the "see context" button visible in the right-hand panel.

8.2. Data Set

The primary dataset used in this study comes from a University of Wisconsin-Milwaukee Anatomy and Physiology class that uses Classroom Salon. This class uses Classroom Salon for hosting the course texts as well as utilizing its functionality as an integral learning aid. The dataset gathered from this Classroom Salon platform is made up of two components: a collection of individual sections from the course textbook [34] and a CSV file containing all of the annotation data that students produced across all the documents. In total, 73 course readings were downloaded as well as 28,020 total student annotations, an average of 384 annotations per course reading (roughly 1 annotation per student in the class).

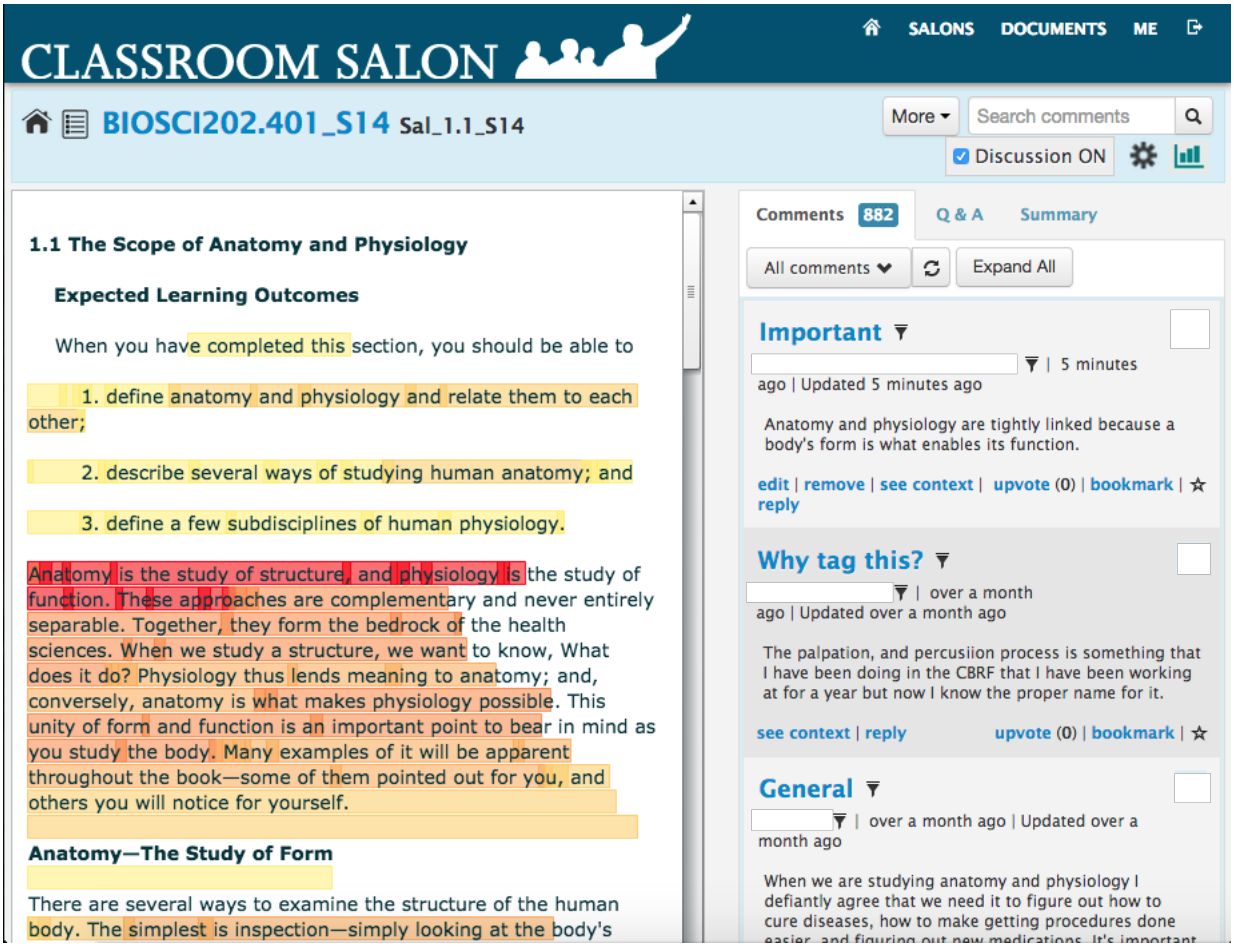

Figure 2: Hotspots in Classroom Salon. Similar to a heat-map, the more red the region, the more heavily annotated it was.

8.3. Choice of Evaluation Procedure

In order to evaluate the performance of Hotspot Summarization, Sentiment-Based Hotspot Summarization, ATMS-S, ATMS-A, and ATMS-Top, I used two different evaluation schemes: Louis & Nenkova's human-free evaluation scheme (*SysSumm*) [21] and the *ROUGE* scheme.

I chose to apply *SysSumm* for one primary reasons: it is cheap. Because *SysSumm* does not require human generated gold-standard summaries, I was able to avoid the difficulty, and the expense, of recruiting individuals (ideally professionals) to manually generate summaries of each document for evaluation purposes. This hurdle would be especially pronounced in this study as I seek to test summarization systems across a substantial number of documents.

I could have sidestepped this problem by using a Document Understanding Conference (DUC)

dataset. The DUC was a conference held every year from 2000-2007 that was geared towards the advancement of automatic summarization. In order to aid this research, the conference gathered large collections of source documents (usually newspapers) and hired professionals to manually generate summaries that could be used for evaluation. Using a DUC dataset would have alleviated the difficulty of gathering manually generated summaries but had the consequence of forcing an abandonment of Classroom Salon's collection of densely annotated documents. The expense required in getting participants to annotate the source documents in the DUC dataset would have been even more costly than gathering a collection of human generated summaries. Due to these practical considerations, the human-free scheme proposed by Louis & Nenkova was very appealing.

Despite the costs involved in the attainment of gold-standard summaries, the *ROUGE* evaluation scheme remains one of the most standard methods of evaluation in the field of automatic summarization. Therefore, I felt that it was necessary to apply a *ROUGE* evaluation, albeit a modified version, to the dataset for the increased validity and academic rigor it provides.

8.4. *SysSumm* Setup

For each of the 73 source documents, summaries were generated using each algorithm in the suite of baselines, in addition to those originating in this study. See Table 4 below for the complete list.

Baseline Methods	Salon Specific Methods
Lead-Sentence	Hotspot
Luhn	Zhang
Edmundson	ATMS-S*
KLSum	ATMS-A*
SumBasic	ATMS-Top*
LexRank	
LSA	

Table 4: The summarizers used in this experiment.
*** Implemented using LSA (5 topics), LDA (5 topics), and HDP**

The baseline methods used in this study correlate very closely, though not perfectly with those used proposed by Louis & Nenkova. One difference is that I replaced their use of the Mead summarizer with an implementation of Edmundson's algorithm. I performed this swap primarily

due to the practical benefit that Edmundson's algorithm was already implemented in *Sumy*. I reasoned that since Edmundson's algorithm makes use of a somewhat similar feature set as Mead (particularly sentence location), this replacement would not lead to serious degradation of *SysSumm*'s validity. Another difference is that in the baseline set, I omitted the Topic Signatures algorithm. Since this study is only concerned with the case of single document summarization, the Topic Signatures algorithm did not readily apply. In its place, I added Luhn's algorithm to the set of baselines. All of the other baselines used by Louis & Nenkova were also used in this study.

For every document in the dataset, a JS Divergence score for each algorithm in Table 4 was calculated by feeding its output summary, in addition to the collection of baseline summaries, into *SysSumm*. All summaries generated contained 10% of the total number of sentences in the document.

8.5. *ROUGE* Setup

In order to determine if the results of the *SysSumm* procedure are legitimate, a more standard *ROUGE* evaluation was performed. As described earlier, the biggest challenge with using *ROUGE* as an evaluation method is its dependence on having at least one, through preferably multiple, human generated reference summaries per document. In order to mitigate the difficulty of acquiring multiple human generated reference summaries for each of the 73 documents, I first selected a random sample of 25 documents over which to perform *ROUGE* analysis.

For each of the 25 documents, volunteers from the Princeton University undergraduate student body generated three reference summaries. Volunteers were provided with a text file containing the source document and were instructed to extract the n most important sentences of the document where n was a value provided to them and equal to 10% of the total number of sentences in the document. Once all reference summaries were created, I used Kavita Ganesan's ROUGE *2.0* [9], a java implementation of the *ROUGE* procedure, for evaluation.

This experimental setup has some limitations of its own. For one, using only a subset of the document pool has the potential of rendering the results unrepresentative of the entire set,

however, by choosing a random sample across the documents we mitigate this as much as possible. Secondly, this study uses volunteers to produce reference summaries opposed to the professionals that are enlisted in more large-scale studies like those performed by the Document Understanding Conference (DUC). This difference is critical to note because much of the literature surrounding automatic summarization uses DUC datasets, thus any direct comparisons with the results of other studies must include the caveat that differences could be a product of different summarizers. Unfortunately, enlisting professional summarizers was not possible for this study. However, I attempt to mitigate the relative inexperience of the volunteer summarizers by gathering multiple reference summaries for each document. *ROUGE* evaluation has been shown to be more accurate when multiple professionally generated summaries are available [37] and I expect the same to hold true when the summaries are produced by amateurs.

9. Results

9.1. *SysSumm* Results

When all of the summarization algorithms (annotation-based and purely text-based) are evaluated using the *SysSumm* procedure, the best performing algorithm (the one with the lowest average JS Divergence score), is ATMS-A (Hdp), which has an average JS Divergence score of 0.147. However, the variation between the algorithms appears to be relatively small having a standard deviation of only 0.041. Figure 3 compares the average JS Divergence score of each summarization system (see Appendix 7 for more complete statistics).

Due to the small standard deviation in JS Divergence scores, I performed an ANOVA test on the dataset to determine if the differences in the average scores of the summarization systems were statistically significant. The differences were in fact statistically significant ($F = 82.66, p < 2 \times 10^{-16}$). Despite telling us that there is a statistically significant difference between the summarization systems, the ANOVA test does not tell us which pairs of systems differ in a statistically significant way. In order to gain insight into these factors, I perform a Tukey Post Hoc Test of significance with the detailed results reported in Table 5. The Tukey HSD test reveals that the best performing

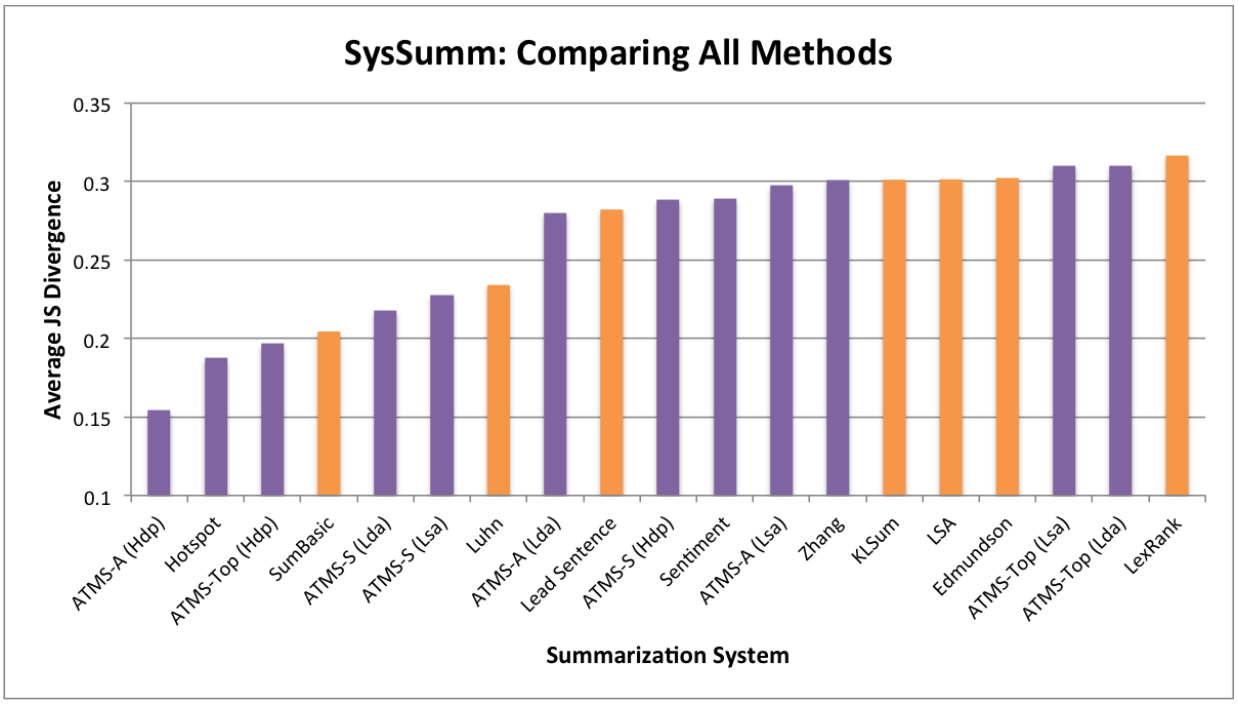

Figure 3: Shows the average JS Divergence score of each summarization algorithm. Purple bars indicate that the algorithm is annotation based while orange bars indicate the algorithm does not rely on annotations.

algorithm, ATMS-A (Hdp), outperforms all the other methods by a statistically significant margin ($p < .05$). The second place algorithm, Hotspot, outperforms all of the remaining algorithms, other than ATMS-Top (Hdp) and SumBasic, by a statistically significant margin ($p < .05$). Furthermore, the top 7 algorithms outperform the bottom 12 at a statistically significant margin ($p < .05$), with 5 of the 7 being methods that utilize the annotation data.

9.2. Discussion of *SysSumm* Results

The *SysSumm* evaluation produces very optimistic results for annotation based summarizers, particularly for ATMS-A (Hdp) which outperforms all other methods, including the competitive baseline systems, by a statistically significant margin ($p < 0.05$). Also, performing well is the Hotspot Summarizer, which outperforms every algorithm other than ATMS-Top (Hdp) and SumBasic by a statistically significant margin. The fact that 5 of the top 7 algorithms are annotation based indicates that annotations provide very valuable data that, when properly extracted, can be effectively used in automatic summarization. Furthermore, the strong performance of the comment-based ATMS-A

Method	Mean JS Divergence	Tukey's HSD
ATMS-A (Hdp)	0.1542	a
Hotspot	0.1878	b
ATMS-Top (Hdp)	0.1966	b c
SumBasic	0.2045	b c d
ATMS-S (Lda)	0.2181	c d e
ATMS-S (Lsa)	0.2280	d e
Luhn	0.2341	e
ATMS-A (Lda)	0.2800	f
Lead Sentence	0.2822	f
ATMS-S (Hdp)	0.2885	f g
Sentiment	0.2893	f g
ATMS-A (Lsa)	0.2974	f g h
Zhang	0.3009	f g h
KLSum	0.3012	f g h
LSA	0.3015	f g h
Edmundson	0.3023	f g h
ATMS-Top (Lsa)	0.3102	g h
ATMS-Top (Lda)	0.3104	g h
LexRank	0.3166	h

Table 5: Methods with the same letter are not significantly different from each other (Tukey HSD test, $p > 0.05$). For example, this table shows that Hotspot and ATMS-Top (Hdp) both have a "b", so they are not significantly different; ATMS-A (Hdp) and Hotspot do not have a letter in common, so they are significantly different.

(Hdp) summarizer and the highlight-based Hotspot summarizer, reveals that both components of an annotation, comments and highlights, are effective tools for summarization in their own right. Future research should be done exploring methods that combine the power of these two features into an even more effective summarization system.

There are some important caveats with regard to the topic modeling algorithms utilized that require mentioning. As explained in Section 6.1.3, the HDP algorithm has the useful property that it automatically infers the number of topics found in a piece of text. LSA and LDA, on the other hand, do not have this ability, thereby requiring a predetermined number of topics for which to build distributions on. For the entirety of this experiment, the number of topics for LSA and LDA was set to 5, which I deemed to be a reasonable choice given the relative length and structure of each document. Despite my best guesses, however, the number of topics chosen was still somewhat arbitrary and the performance of the ATMS algorithms using LSA and LDA as their topic modeling

algorithms could change as the number of prescribed topics varies. Further research could build upon these results by allowing the number of topics to vary for LSA and LDA.

9.3. *ROUGE* Results

When all of the summarization algorithms (annotation based and purely text based) are evaluated using the *ROUGE*-2 procedure, the best performing algorithm (the one with the highest average recall score), is the simple Hotspot Summarizer, which has an average recall score of 0.3049. However, the variation between the algorithms appears to be relatively small having a standard deviation of only 0.0556. Figure 4 compares the average recall score of each summarization system (see Appendix 8 for more complete statistics).

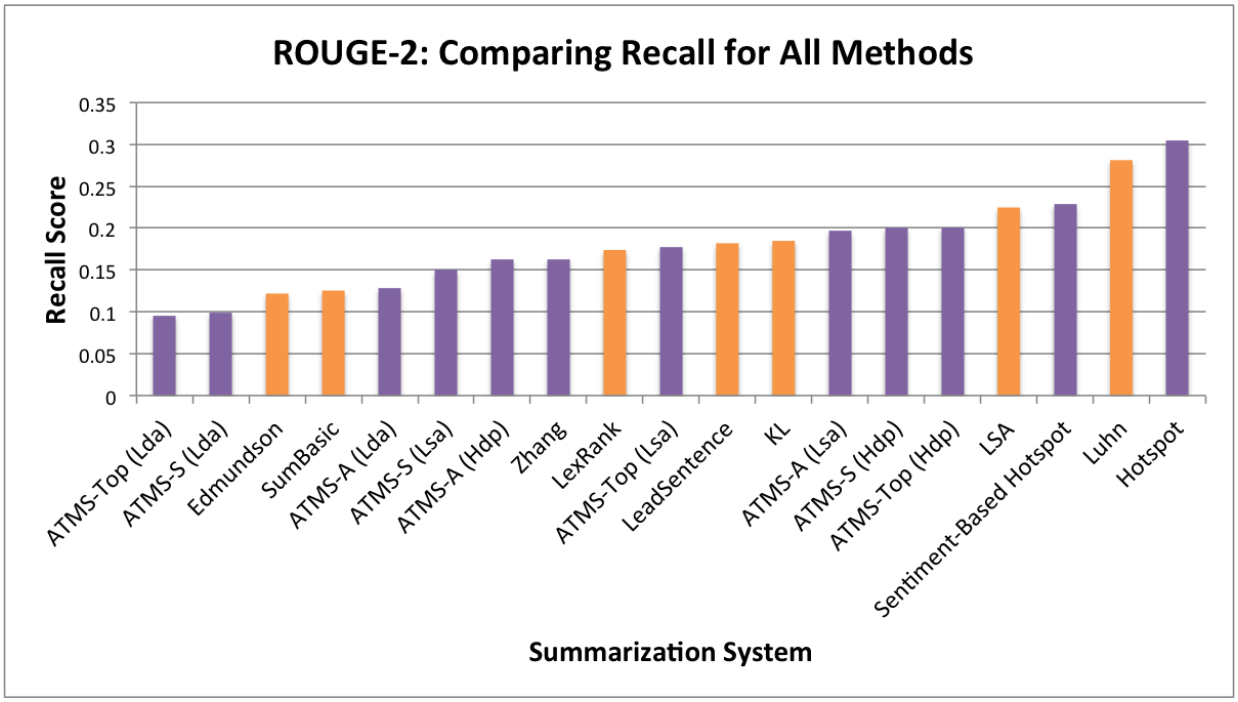

Figure 4: Shows the average recall score of each summarization algorithm. Purple bars indicate that the algorithm is annotation based while orange bars indicate the algorithm does not rely on annotations.

Due to the very small standard deviation of recall scores, I performed an ANOVA test across the dataset to determine if the differences in the average scores of the summarization systems were statistically significant. The differences were in fact statistically significant ($F = 10.86, p < 2 \times 10^{-16}$).

Despite telling us that there is a statistically significant difference between the summarization systems, the ANOVA test does not tell us which pairs of systems differ in a statistically significant way. In order to gain insight into these factors, I perform a Tukey Post Hoc Test of significance with the detailed results reported in Table 6. The Tukey HSD test reveals that there is no significant difference in the recall scores amongst the top four algorithms: Hotspot, Luhn, Sentiment-Based Hotpot, and LSA. Despite this lack of statistically significant differential amongst the top algorithms, all algorithms outside of the top four are shown to perform worse than the Hotspot Summarizer at a statistically significant level.

The three ATMS algorithms with the highest mean recall score are ATMS-Top (Hdp), ATMS-S (Hdp), and ATMS-A (Lsa). These three methods perform at a level that is statistically indistinguishable from all of the baseline algorithms tested at the 95% confidence level. The worst performing algorithms are ATMS-S (Lda) and ATMS-Top (Lda) which had mean recall scores of 0.0987 and 0.0955 respectively, which are significantly worse than the top performing annotation based algorithms.

Since a Turkey HSD test is a conservative significance test for multiple comparisons, many groups that might in fact be significantly different from one another, may fail to be reported by this test. However, by first performing the Tukey HSD, we can more confidently group algorithmic methods into tiers which allows us to mitigate the probability of obtaining a Type 1 error while still performing more liberal statistical comparisons. One comparison that is of particular interest is between the Hotspot and Sentiment Based Hotspot summarizers. The conservative Turkey HSD reports no significant difference between these two methods, however the more powerful paired t-test reports that the mean difference between the groups of 0.0765 is statistically significant ($t = 2.9486, df = 24, p < 0.01$).

9.4. Discussion of *ROUGE* Results

The biggest difficulty with analyzing the results coming from this evaluation procedure is the lack of statistically significant difference between the performance of most algorithms. In some instances,

Method	Mean Recall	Tukey's HSD
Hotspot	0.3049	a
Luhn	0.2813	a b
Sentiment	0.2284	a b c
Lsa	0.2247	a b c
ATMS-Top (Hdp)	0.2012	b c d
ATMS-S (Hdp)	0.2003	b c d
ATMS-A (Lsa)	0.1968	b c d
KLSum	0.1848	c d
Lead Sentence	0.1819	c d e
ATMS-Top (Lsa)	0.1779	c d e f
LexRank	0.1739	c d e f
Zhang	0.1630	c d e f
ATMS-A (Hdp)	0.1628	c d e f
ATMS-S (Lsa)	0.1500	c d e f
ATMS-A (Lda)	0.1278	d e f
SumBasic	0.1254	d e f
Edmundson	0.1218	d e f
ATMS-S (Lda)	0.0987	e f
ATMS-Top (Lda)	0.0955	f

Table 6: Methods with the same letter are not significantly different from each other (Tukey HSD test, $p > 0.05$). For example, this table shows that Hotspot and Luhn both have an "a", so they are not significantly different; Hotspot and ATMS-Top (Hdp) do not have a letter in common, so they are significantly different.

this could be due to algorithms actually performing at about the same level of effectiveness. However, the bigger culprit is almost certainly this experiment's low sample size of only 25 documents. At over $1/3$ of the size of the full dataset, I had hoped that 25 documents would be enough from which to draw conclusive evidence from, however it appears as though this was not the case. Unfortunately, the practical restraints explained in Section 8.5 prevented us from increasing this number.

Despite that caveat, the results provide strong evidence that the Hotspot Summarization algorithm is indeed a very effective method for automatically generating summaries as it was in the top tier of algorithms, actually having the highest minimum, maximum, median, and mean recall scores (0.15, 0.63, 0.29, 0.30). On the surface, this result appears obvious – of course highlighted regions are the best candidates for a summary. Prior to this study, however, that assumption would be unfounded because not all annotations are necessarily made to indicate the importance of a textual region. Rather, annotations can indicate a multitude of possibilities including: confusion, general interest,

desired clarification, etc. However, the results indicate that despite the various motivating factors behind any particular annotation, when taken in aggregate, they correlate highly with summary candidate regions. This result is hugely important for Classroom Salon as it provides empirical evidence validating the aggregate power of their trove of annotations.

Another point of interest is the statistically significant margin by which the simple Hotspot Summarizer outperforms the Sentiment-Based Hotspot Summarizer. This result suggests that the sentiment of an annotation acts as noise that occludes the value that the very presence of annotation provides. In other words, it implies that the presence of a highlight is a better predictor of important than the reason behind the annotation. This conclusion is subject to many other factors that warrant future research. For example, the Sentiment Based Hotspot Summarizer is dependent upon SentiWordNet's ability to accurately provide positivity and negativity scores of each word in the annotation. If these values are not accurate, then neither are the results, thus motivating further research into the validity of SentiWordNet. Also, though the algorithm assigns each annotation a continuous value representing its positivity and negativity score, it is possible that a binary classification would perform better.

Due to a lack of statistical significance, there is not much that can be said with regards to the ATMS algorithms. With the possible exception of ATMS-S (Lda) and ATMS-Top (Lda), the ATMS algorithms perform at levels not significantly different from all of the baseline algorithms when measured by *ROUGE*-2 recall scores. Future research that can expand this experimental procedure across a larger collection of texts is necessary in order to concretely determine the effectiveness of these approaches.

9.5. Discussion on Differences in Evaluation Procedures

It is difficult to determine how concordant the results of the *SysSumm* and *ROUGE* evaluations are. Research done by Louis & Nenkova suggests that the results of both systems should be relatively similar, however the lack of power provided by this experiment's *ROUGE* evaluation makes it impossible to either validate or invalidate this relationship on the dataset. Future work

that is able to reproduce this study, while incorporating a manual evaluation procedure such as the Pyramid method, would help to provide insights into how *SysSumm* and *ROUGE* compare to manual evaluation over this dataset.

10. Conclusion

Through my research I have determined many important avenues for future investigation. First and foremost, future research that duplicates this study's methodology, while expanding it to cover a much larger document set would yield vital data regarding the performance of the ATMS algorithms relative to each other and to the other baseline metrics in existence. This research would help to uncover concrete evidence either supporting or rejecting the effective application of the ATMS algorithms to the summarization task.

In addition, future research needs to be done exploring the performance of the different summarization algorithms when the length of the produced summaries vary. In this study I only created summaries of length equal to 10% of the total number of document sentences which is about 2.5$\times$ the average percentage of sentences annotated by a user. An interesting avenue of future research would look into how the proposed algorithms fair when creating summaries that are both closer to, and further from, this per-person average percentage.

A particularly novel area of future study would involve methods for creating personalized summaries. Platforms like Classroom Salon are very powerful in their ability to aggregate annotations across hundreds of students; a capability that this study made use of. With this study demonstrating the effectiveness of using hotspots for summarization, a natural extension would be to determine if an individual's unique annotating behavior can be used to generate user-specific hotspots in texts, a priori.

Through this study, I have presented a suite of novel approaches to automatically generating summaries. These approaches all make use of user annotations related to the source text. The ATMS algorithms all rely on finding topic similarity between the source text and the text's annotations while the Hotspot and Sentiment-Based Hotspot methods are primarily based upon analyzing

which regions of the text are annotated. The result of applying these methods for single document, extractive summarization is cautiously optimistic. Through *SysSumm* evaluation I have determined that ATMS algorithms, particularly ATMS-A (Hdp), as well as the Hotspot Summarizer are capable of outperforming nearly all-competitive baselines. This result lends empirical evidence to the intuitive value of the annotations and paves the way for future research that builds off of these approaches. Unfortunately, this study was not able to reaffirm these findings with a *ROUGE* evaluation due to the lack of statistical power present in the rankings it produced. Finally, I have added implementations of my proposed algorithms, as well as some powerful baseline methods to the open-source summarization package *Sumy*, enabling future researchers to build from where this study leaves off. Details can be found at https://github.com/JulianGriggs/sumy.

References

[1] *LexRank: Graph-based Lexical Centrality as Salience in Text Summarization*, vol. 22, 2004.

[2] D. Austin, "How google finds your needle in the web's haystack," *American Mathematical Society*, December 2006.

[3] R. Barzilay and M. Elhadad, "Using lexical chains for text summarization," in *In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, 1997, pp. 10–17.

[4] M. Belica. sumy. [Online]. Available: https://github.com/miso-belica/sumy

[5] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, 2012.

[6] J. M. Conroy and D. P. O'leary, "Text summarization via hidden markov models," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '01. New York, NY, USA: ACM, 2001, pp. 406–407. [Online]. Available: http://doi.acm.org/10.1145/383952.384042

[7] H. P. Edmundson, "New methods in automatic extracting," *J. ACM*, vol. 16, no. 2, pp. 264–285, Apr. 1969. [Online]. Available: http://doi.acm.org/10.1145/321510.321519

[8] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining," in *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06*, 2006, pp. 417–422.

[9] K. Ganesan. Rouge 2.0 - java package for evaluation of summarization tasks with updated rouge measures. [Online]. Available: http://kavita-ganesan.com/content/rouge-2.0

[10] Z. Ghahramani, "Hidden markov models." River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2002, ch. An Introduction to Hidden Markov Models and Bayesian Networks, pp. 9–42. [Online]. Available: http://dl.acm.org/citation.cfm?id=505741.505743

[11] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '01. New York, NY, USA: ACM, 2001, pp. 19–25. [Online]. Available: http://doi.acm.org/10.1145/383952.383955

[12] 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*, ser. NAACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 362–370. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620754.1620807

[13] M. A. Hearst, "Multi-paragraph segmentation of expository text," in *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ser. ACL '94. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994, pp. 9–16. [Online]. Available: http://dx.doi.org/10.3115/981732.981734

[14] G. Heinrich, ""infinite lda" - implementing the hdp with minimum code complexity," February 2011, technical note TN2011/1.

[15] M. Hu, A. Sun, and E.-P. Lim, "Comments-oriented document summarization: Understanding documents with readers' feedback," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '08. New York, NY, USA: ACM, 2008, pp. 291–298. [Online]. Available: http://doi.acm.org/10.1145/1390334.1390385

[16] J. S. Karel Ježek, "Automatic text summarization (the state of the art 2007 and new challenges)." [Online]. Available: http://textmining.zcu.cz/publications/Z08.pdf

[17] S. B. . E. L. . E. Klein, *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[18] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '95. New York, NY, USA: ACM, 1995, pp. 68–73. [Online]. Available: http://doi.acm.org/10.1145/215206.215333

[19] C.-Y. Lin and E. Hovy, "The automated acquisition of topic signatures for text summarization," in *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, ser. COLING '00. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 495–501. [Online]. Available: http://dx.doi.org/10.3115/990820.990892

[20] ——, "Manual and automatic evaluation of summaries," in *Proceedings of the Workshop on Automatic Summarization*. ACL-02, July 7-12 2002.

[21] A. Louis and A. Nenkova, "Automatically assessing machine summary content without a gold standard," *Comput. Linguist.*, vol. 39, no. 2, pp. 267–300, Jun. 2013. [Online]. Available: http://dx.doi.org/10.1162/COLI_a_00123

[22] H. P. Luhn, "The automatic creation of literature abstracts," *IBM J. Res. Dev.*, vol. 2, no. 2, pp. 159–165, Apr. 1958. [Online]. Available: http://dx.doi.org/10.1147/rd.22.0159

[23] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*, online ed. Cambridge University Press, 2009. [Online]. Available: http://nlp.stanford.edu/IR-book/html/htmledition/dot-products-1.html

[24] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," in *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[25] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995. [Online]. Available: http://doi.acm.org/10.1145/219717.219748

[26] R. Moro and M. Bielikov', "Personalized text summarization based on important terms identification," in *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*, Sept 2012, pp. 131–135.

[27] J. Morris and G. Hirst, "Lexical cohesion computed by thesaural relations as an indicator of the structure of text," *Comput. Linguist.*, vol. 17, no. 1, pp. 21–48, Mar. 1991. [Online]. Available: http://dl.acm.org/citation.cfm?id=971738.971740

[28] A. Nenkova and K. McKeown, "Automatic summarization," *Foundations and Trends in Information Retrieval*, vol. 5, pp. 103–233, 2011.

[29] A. Nenkova and R. J. Passonneau, "Evaluating content selection in summarization: The pyramid method," *North American Chapter of the Association for Computational Linguistics - NAACL*, pp. 145–152, 2004.

[30] A. Nenkova and L. Vanderwende, "The impact of frequency on summarization," Microsoft Research, Tech. Rep., 2005.

[31] D. R. Radev, H. Jing, M. Styś, and D. Tam, "Centroid-based summarization of multiple documents," *Inf. Process. Manage.*, vol. 40, no. 6, pp. 919–938, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.ipm.2003.10.006

[32] G. J. Rath, A. Resnick, and T. R. Savage, "The formation of abstracts by the selection of sentences. part i. sentence selection by men and machines," *American Documentation*, vol. 12, no. 2, pp. 139–141, 1961. [Online]. Available: http://dx.doi.org/10.1002/asi.5090120210

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

[34] K. Saladin, *Anatomy & Physiology: The Unity of Form and Function*, 6th ed. McGraw Hill Higher Education, 2011.

[35] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the American Statistical Association*, vol. 101, 2004.

[36] D. L. Thomas K. Landauer, Peter W. Foltz, "An introduction to latent semantic analysis," *Discourse Processes*, vol. 25, pp. 259–284, 1998.

[37] C. yew Lin, "Rouge: a package for automatic evaluation of summaries," 2004, pp. 25–26.

[38] H. Zhang, "A study for document summarization based on personal annotation," in *In HLT-NAACL 2003 Workshop: Text Summarization (DUC03*, 2003.

11. Appendix

Algorithm 1 Basic Algorithm for Extracting Sentences

Require: D =document; N =number of words in D; S = sentence; W =word

calculateWordProbabilities
$st \leftarrow dictionary$[word, probability]
for all Words(W) in Document(D) **do**
  if st contains(W) **then**
    $st[W] += 1/N$
  else
    $st[W] = 1/N$
  end if
end for
return st

calculateSentenceProbabilities
$st_{word} \leftarrow calculateWordProbabilities$
$st_{sentence} \leftarrow dictionary$[sentence, probability]
for all Sentences(S) in Document(D) **do**
  $wordsInS \leftarrow 0$
  for all Words(W) in Sentence(S) **do**
    $st_{sentence}[S] = st_{sentence}[S] + st_{word}[W]$
    $wordsInS = wordsInS + 1$
  end for
  $st_{sentence}[S] = st_{sentence}[S]/wordsInS$
end for
return $st_{sentence}$

extractSentences
$st_{sentence} \leftarrow dictionary$[sentence, probability]
$Summary =$ ""
while Summary Length < Desired Summary Length **do**
  Add $max(st_{sentence})$ to $Summary$
  Remove $max(st_{sentence})$ from $st_{sentence}$
end while
return $Summary$

Algorithm 2 ATMS-A pseudo code

Require: *words* = List of unique words in the text; *sentences* = List of sentences in the text;
comments = List of Comments; L = Length of desired summary; *tm* = topic modeling algorithm

---

wordXComment ← create (word x comment) frequency matrix(*words*, *comments*)
$topicDist \leftarrow tm(wordsXComment)$
$commentsInTopicSpace \leftarrow$ topic space transform($topicDist, wordXComment$)
aggregateSimilarity ← map all *s* in *sentences* to 0
for all s in sentences **do**
  sCommentWordFreq ← Transform *s* into *comments* vector space
  $sInTopicSpace \leftarrow$ topic space transform($topicDist, sCommentWordFreq$)
  for all *c* in *comments* **do**
    $aggregateSimilarity[s] +=$ cosineDistance($commentsInTopicSpace[c], sInTopicSpace$)
  end for
end for
sortedMostSimilar ← sort high to low by value(*aggregateSimilarity*)
summary ← top L sentences in *sortedMostSimilar*

Algorithm 3 ATMS-S pseudo code

Require: *words* = List of unique words in the text; *sentences* = List of sentences in the text;
comments = List of Comments; L = Length of desired summary; *tm* = topic modeling algorithm

---

wordXSent ← create (word x sentence) frequency matrix(*words*, *sentences*)
$topicDist \leftarrow tm(wordsXSent)$
$sentencesInTopicSpace \leftarrow$ topic space transform($topicDist, wordXSent$)
aggregateSimilarity ← map all *s* in *sentences* to 0
for all c in *comments* **do**
  cDocWordFreq ← Transform *c* into *words* vector space
  $cInTopicSpace \leftarrow$ topic space transform($topicDist, cDocWordFreq$)
  for all *s* in *sentences* **do**
    $aggregateSimilarity[s] +=$ cosineDistance($sentencesInTopicSpace[s], cInTopicSpace$)
  end for
end for
sortedMostSimilar ← sort high to low by value(*aggregateSimilarity*)
summary ← top L sentences in *sortedMostSimilar*

Algorithm 4 ATMS-Top preprocessing

Require: $words$ = List of unique words in the text; $sentences$ = List of sentences in the text; $comments$ = List of Comments; K = Number of desired comments; tm = topic modeling algorithm

---

$wordXComment \leftarrow$ create (word x comment) frequency matrix$(words, comments)$
$topicDist \leftarrow tm(wordsXComment)$
$commentsInTopicSpace \leftarrow$ topic space transform$(topicDist, wordXComment)$
$aggregateSimilarity \leftarrow$ map all c in $comments$ to 0
for all c0 in comments **do**
  $cCommentWordFreq \leftarrow$ Transform $c0$ into $comments$ vector space
  $cInTopicSpace \leftarrow$ topic space transform$(topicDist, cCommentWordFreq)$
  for all $c1$ in $comments$ **do**
    $aggregateSimilarity[c0] +=$ cosineDistance$(commentsInTopicSpace[c1], cInTopicSpace)$
  end for
end for
$sortedMostSimilar \leftarrow$ sort high to low by value$(aggregateSimilarity)$
$summary \leftarrow$ top K comments in $sortedMostSimilar$

Method	Minimum	Maximum	Median	Mean
ATMS-A (Hdp)	0.1111	0.2633	0.1476	0.1542
Hotspot	0.1302	0.2760	0.1830	0.1878
ATMS-Top (Hdp)	0.1314	0.2906	0.1956	0.1966
SumBasic	0.1298	0.3731	0.1996	0.2045
ATMS-S (Lda)	0.1304	0.3249	0.2141	0.2181
ATMS-S (Lsa)	0.1450	0.3942	0.2211	0.2280
Luhn	0.1500	0.3675	0.2311	0.2341
ATMS-A (Lda)	0.1865	0.4266	0.2768	0.2800
Lead Sentence	0.1946	0.4315	0.2710	0.2822
ATMS-S (Hdp)	0.1654	0.4079	0.2789	0.2885
Sentiment	0.2056	0.4487	0.2826	0.2893
ATMS-A (Lsa)	0.1977	0.4535	0.2942	0.2974
Zhang	0.1918	0.4891	0.3000	0.3009
KLSum	0.2063	0.4112	0.3013	0.3012
LSA	0.2057	0.4668	0.2964	0.3015
Edmundson	0.2064	0.4392	0.2954	0.3023
ATMS-Top (Lsa)	0.2267	0.4579	0.3140	0.3102
ATMS-Top (Lda)	0.2165	0.4368	0.3042	0.3104
LexRank	0.2175	0.4594	0.3129	0.3166

Table 7: *SysSumm*: Shows the minimum, maximum, median, and mean JS Divergence scores of each summarization algorithm. The cells highlighted in yellow are the minimum (best) values for that metric.

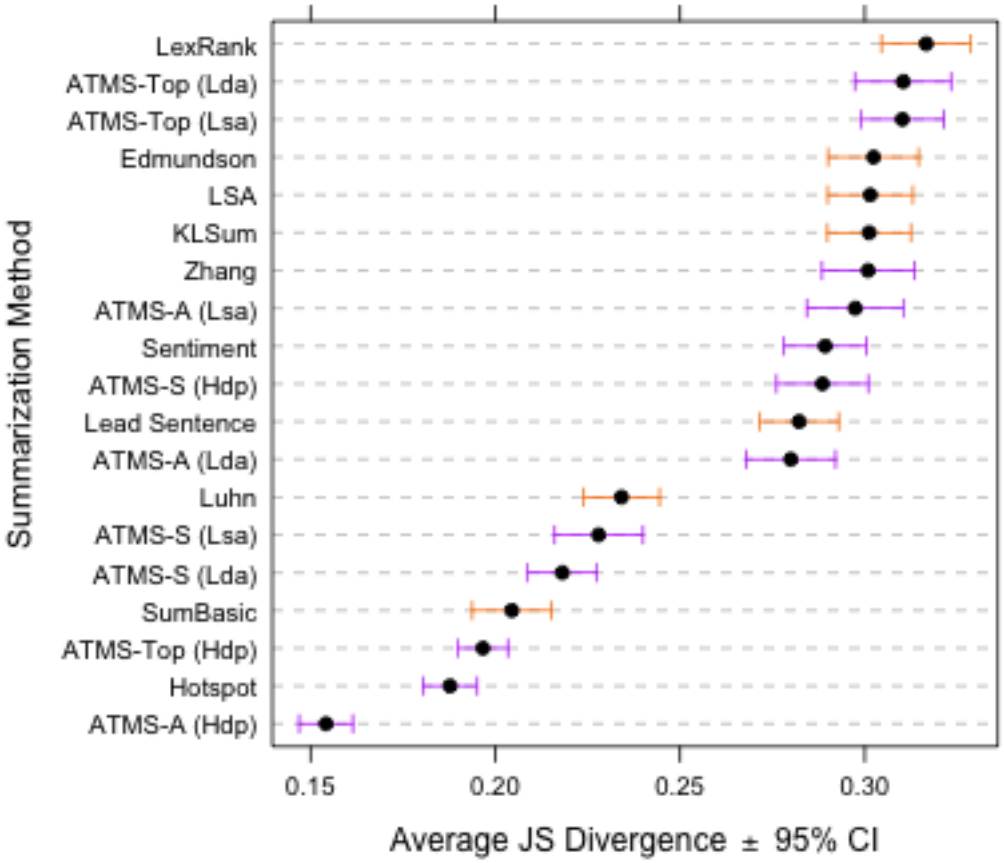

Figure 5: *SysSumm*: This figure shows the confidence intervals for the average JS Divergence score for each method. Methods with overlapping intervals are not significantly different from one another ($p > 0.05$)

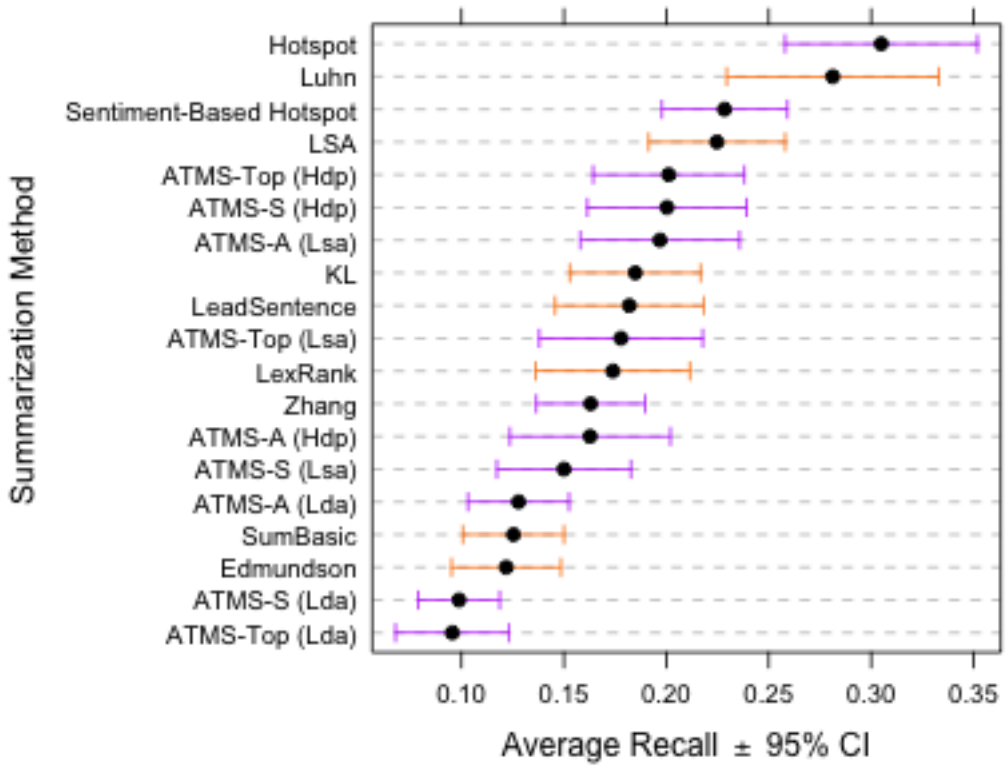

Figure 6: *ROUGE*: This figure shows the confidence intervals for the average recall score for each method. Methods with overlapping intervals are not significantly different from one another ($p > 0.05$)

Method	Minimum	Maximum	Median	Mean
Hotspot	0.15034	0.63315	0.2922	0.3049368
Luhn	0.02182	0.62303	0.28912	0.2812548
Sentiment-Based Hotspot	0.07485	0.38946	0.22641	0.2284276
LSA	0.09129	0.39376	0.20835	0.2246812
ATMS-Top (Hdp)	0.00973	0.40476	0.1851	0.2011784
ATMS-S (Hdp)	0.03661	0.35777	0.20607	0.2003096
ATMS-A (Lsa)	0.05384	0.40506	0.19109	0.1968928
KL	0.04352	0.44051	0.18313	0.1848248
Lead Sentence	0.02385	0.35755	0.16454	0.1819352
ATMS-Top (Lsa)	0.03184	0.42463	0.18407	0.1778532
LexRank	0	0.36025	0.17575	0.1738856
Zhang	0.06763	0.28936	0.14295	0.1630304
ATMS-A (Hdp)	0.00853	0.36993	0.15432	0.1627896
ATMS-S (Lsa)	0.05195	0.42463	0.14336	0.1500124
ATMS-A (Lda)	0.03278	0.21041	0.13287	0.1278432
SumBasic	0.03931	0.24297	0.12339	0.1253624
Edmundson	0.01682	0.25481	0.11772	0.1218372
ATMS-S (Lda)	0.03571	0.24001	0.08763	0.0987488
ATMS-Top (Lda)	0.02385	0.27435	0.06511	0.095528

Table 8: *ROUGE*: Shows the minimum, maximum, median, and mean recall scores of each summarization algorithm. The cells highlighted in yellow are the maximum (best) values for that metric.

1.1 The Scope of Anatomy and Physiology	10.1 The Structural and Functional Organization of Muscles
1.2 The Origins of Biomedical Science	10.2 Muscles of the Head and Neck
1.3 Scientific Method	10.4 Muscles Acting on the Shoulder and Upper Limb
1.4 Human Origins and Adaptations	10.5 Muscles Acting on the Hip and Lower Limb
1.5 Human Structure	11.1 Types and Characteristics of Muscular Tissue
1.6 Human Function	11.2 Microscopic Anatomy of Skeletal Muscle
1.7 The Language of Medicine	11.3 The Nerve?Muscle Relationship
1.8 Review of Major Themes	11.4 Behavior of Skeletal Muscle Fibers
2.1 Atoms Ions and Molecules	11.5 Behavior of Whole Muscles
2.2 Water and Mixtures	11.6 Muscle Metabolism
2.3 Energy and Chemical Reactions	12.1 Overview of the Nervous System
2.4 Organic Compounds	12.2 Properties of Neurons
3.1 Concepts of Cellular Structure	12.3 Supportive Cells (Neuroglia)
3.2 The Cell Surface	12.4 Electrophysiology of Neurons
3.4 The Cell Interior	12.5 Synapses
4.1 DNA and RNA?The Nucleic Acids	12.6 Neural Integration
4.2 Genes and Their Action	13.1 The Spinal Cord
4.3 DNA Replication and the Cell Cycle	13.2 The Spinal Nerves
4.4 Chromosomes and Heredity	13.3 Somatic Reflexes
5.1 The Study of Tissues	14.1 Overview of the Brain
5.2 Epithelial Tissue	14.2 Meninges Ventricles Cerebrospinal Fluid and Blood Supply
5.3 Connective Tissue	14.3 The Hindbrain and Midbrain
5.4 Nervous and Muscular Tissues?Excitable Tissues	14.5 Integrative Functions of the Brain
5.5 Cell Junctions Glands and Membranes	14.6 The Cranial Nerves
5.6 Tissue Growth Development Repair and Degeneration	15.1 General Properties of the Autonomic Nervous System
6.1 The Skin and Subcutaneous Tissue	15.2 Anatomy of the Autonomic Nervous System
6.2 Hair and Nails	15.3 Autonomic Effects on Target Organs
6.3 Cutaneous Glands	5.4 Central Control of Autonomic Function
6.4 Skin Disorders	16.1 Properties and Types of Sensory Receptors
7.1 Tissues and Organs of the Skeletal System	16.2 The General Senses
7.2 Histology of Osseous Tissue	16.3 The Chemical Senses
7.3 Bone Development	16.4 Hearing and Equilibrium
7.4 Physiology of Osseous Tissue	16.5 Vision
7.5 Bone Disorders	17.1 Overview of the Endocrine System
9.1 Joints and Their Classification	17.2 The Hypothalamus and Pituitary Gland
9.2 Synovial Joints	17.4 Hormones and Their Actions
9.3 Anatomy of Selected Diarthroses	

Table 9: This table lists all of the titles of each document in our dataset. Each title is a subsection of a textbook on Anatomy and Physiology. Note: Some sections of the textbook are missing from this list due to acquisition difficulties. and were therefore omitted from the dataset.