

Evaluation of LSA and TextRank Methods for Automatic Text Summarization ^{*}

Johnny Sellers [†]

April 11, 2019

Abstract

Automatic text summarization is an efficient and valuable data-driven technique for contending with the vast amount of textual data in existence. This report covers an experiment in which TED Talk transcripts were analyzed using unsupervised, extractive techniques for automatic text summarization—Latent Semantic Analysis (LSA) and TextRank. Both techniques performed similarly on the data set based on the ROUGE-N, ROUGE-L and cosine similarity evaluation metrics. The LSA-based techniques generated qualitatively better summaries than the TextRank approach in a specific document case study (see §4.1). This was ultimately due to the differing feature extraction processes of the methods: the LSA approaches employed a semi-supervised concept derivation process while TextRank employed a purely unsupervised one largely affected by both the target summary and document lengths. In the specific document case study, increasing the target summary length from three to seven enabled the TextRank algorithm to generate a summary that captured the most important points of the talk arguably better than the LSA-based methods.

1 Introduction and Overview

Text mining is the process of deriving high-quality information from text. Tasks aimed at this endeavor include information retrieval, lexical analysis, pattern recognition, text clustering, summarization, sentiment analysis, and so on. These all belong under the heading of

^{*}This paper was originally submitted as a final project for Dr. J.N. Kutz’s course, *AMATH 582: Computational Methods for Data Analysis*, at the University of Washington, Seattle, WA, Winter 2019 quarter.

[†]Student, Applied Mathematics Department, University of Washington, Seattle, WA

natural language processing. With vast amounts of textual data in existence, automatic summarization is a significant tool of NLP for doing efficient analyses of text to glean significant information. Today, automatic text summarization is used in situations such as automated call systems, image labeling, and most notably search engines.

Methods for text summarization fall into two categories: extractive and abstractive. The latter is based on the idea of deriving meaning from text to generate original summaries instead of selecting a compilation of sentences from the document as the former does. In the past, abstractive techniques did not perform as well as extractive methods, but as of late, with deep learning methods, their results have surpassed the extractive, data-driven methods in the quality of summaries generated.

In this experiment, TED Talk transcripts were analyzed by deriving the main concepts and summarizing the important points of the talks. The summaries were generated automatically using one of two extractive, unsupervised methods—*Latent Semantic Analysis* (LSA) and the graph-based algorithm, *TextRank*. The quality of the summaries generated by the two methods were compared using the ROUGE-N, ROUGE-L and cosine similarity evaluation metrics outlined in [9] and [7] respectively. These evaluation measures compare the generated summaries against an *ideal* summary (also known as a *golden summary*) of the document. In one test, evaluation scores were averaged over 50 summaries of TED Talk transcripts where the ideal was taken to be the descriptions of the talks given in the raw data. In a another text, human generated summaries were produced for the first five data records and taken as the ideal. Various parameters including transcript length (i.e., the number of sentences in the transcript), the target summary length, method modifications such as removing stop words, stemming words, and so on were examined for their affect on the evaluation scores.

2 Theoretical Background

Natural language processing (NLP) encompasses many tasks surrounding textual analysis—e.g., character, syntactical, and sentiment analyses, categorization, concept extraction, entity relation modeling, and so on. The objective of this experiment was to analyze text data through *document summarization*, also known as *automatic text summarization*. As the name suggests, this NLP task is carried out at the document level, as opposed to character, word, or sentence level. The earliest attempts at text summarization examined *word* and *phrase frequency*, *position* in the text, and *key phrases* as features for analysis [4] (cf., [10]).

2.1 Automatic Text Summarization

A definition of automatic text summarization is as follows:

Given a text document of length l (i.e., containing l sentences) as input, produce computationally a text having length n shorter than l that conveys the most important information contained in the original text.

Note that the "most important" information depends on the perspective of the analyst(s).

Techniques to automatically create such summaries from a given input document do so by essentially creating an intermediate representation of the input and finding the most important content from this representation. The task of this report was most accurately characterized as *single-document summarization* (as opposed to *multi-document summarization*) since the goal was to produce a summarization from a single document, a TED Talk transcript.

Methods for producing text summaries are broadly based on one of two approaches, *extractive* or *abstractive* summarization [5]; however, some methods make use of both. The latter refers to approaches that attempt to generate a summary from the "bottom up," consisting of sentences not found within the text itself. The extractive approach attempts to generate a summary constructed of sentences found within the document.

2.1.1 Extractive Text Summarization

Basically, extractive summarization techniques choose a subset of the sentences in a document to generate a summary. All of them do so by performing three independent tasks: (1) construct an intermediate representation of the input that best yields the features of the data, (2) score the sentences based on this representation, and (3) construct a summary from the scored sentences. See [7] and [4] for overviews of the various types of extractive approaches for summarization. The approaches taken in this report were LSA- and graph-based.

Intermediate Representation of the Input Document

Constructing the intermediate representation can be done by a *topic representation* or *indicator representation* approach. With indicator representation, every sentence is described as a list of indicators of importance such as the sentence length, sequential position in the document, etc. Topic representation approaches construct the intermediate representation by interpreting the topic(s) discussed in the text. These representation approaches vary by model and complexity and fall into four categories—frequency-driven approach, topic word

approach, latent semantic analysis, and Bayesian topic models. [1]

Sentence Scoring

Once the intermediate representation is constructed, extractive methods score or rank each sentence, in one way or another, by importance. With topic representations, the sentences are ranked by a measure of how they signal the important concepts of the document. The act of obtaining the most pertinent concepts can be done in a supervised or unsupervised fashion. Illustrating this, the latent semantic analysis of this report was carried out by using the tag fields of the TED Talk data to represent concepts—a semi-supervised approach (see §2.2). The TextRank algorithm essentially derives the concept features when it selects the n most similar sentences, resembling a clustering task.

Sentence Selection

Selecting the n most important sentences can be carried out by an extractive method through one of three approaches: (1) The *Best n* approach chooses the n top-ranked sentences in a linear or independent fashion. (2) *Maximal marginal relevance* chooses in an iterative greedy procedure where a sentence is chosen and then the remaining sentence scores are recomputed so that sentences that are too similar to ones already chosen are avoided. (3) *Global selection* aims to optimally choose a collection of n sentences that maximize importance, minimize redundancy, and maximize coherence. [1] The choice of approach impacts how the representation and scoring steps of the method are carried out.

2.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is an unsupervised technique to derive an implicit representation based on co-occurrence of words. [1] The method derives the *latent semantic structure* of the document by reducing the input representation to a linearly independent basis that expresses the main topics. It does so through use of the singular value decomposition (SVD) which captures the interrelationships among terms so that sentences can be clustered based on semantics or context rather than on the basis of words. [7]

2.2.1 Topic Representation

The topic representation (i.e., the input matrix X) of this method is constructed by letting the rows of X represent the main concepts (words taken from the document) and the columns represent sentences of the input. The entries x_{ij} of X are the weight of the concept i in sentence j , where the weight is set to zero if the sentence does not contain the word and

to the textitterm-frequency inverse document-frequency (TF-IDF) of the concept otherwise. The TF-IDF of concept c (a single word) for sentence i is given by

$$\text{TF-IDF}_i(c) = \text{TF}_i(c) \cdot \log_2(\text{DF}(c)), \quad (1)$$

where $\text{TF}(c)$ is the number of times the word c occurs in the sentence i divided by the total number of words in sentence i , and $\text{DF}(c)$ is the total number of sentences in the document divided by the number of sentences containing the word c . The term frequency $\text{TF}(c)$ measures how frequently a word occurs in a sentence, while the document frequency $\text{DF}(c)$ measures the importance of c in the entire document.

An interpretation of the singular value decomposition (SVD) of the m by n input matrix X , given by $X = U\Sigma V^T$, is as follows: The matrix U is a m by n matrix whose columns can be interpreted as topics. The entries of Σ correspond to the weight, or prominence, of each of the topics. Matrix V^T is a new representation of the sentences in terms of the concepts given in U , and the matrix ΣV^T indicates to what extent the sentence conveys the concept, so $\sigma_{ij}v_i$ indicates the weight for concept j in sentence i . [1]

2.2.2 Sentence Scoring and Selection

The sentence selection step of the LSA can take various approaches; see [16] for an overview. Two approaches were evaluated in this experiment—the *cross* method proposed by Ozsoy et al. in [16] and the method of Gong and Liu from [6].

In Gong and Liu’s method, the rows of the matrix of right-singular vectors of X , V^T , represent the concepts of the document while its columns represent the sentences. The concepts are ordered by importance in a decreasing manner; the order can be obtained through manual means or, as in this experiment, by word count. One sentence is selected from each row starting with the first concept until the desired summary length or the end of the matrix is reached. The column (sentence) with the greatest TF-IDF value in the row is the one selected.

The cross method incorporates a pre-processing step and a method for sentence selection based on modified column sums of V^T . The average TF-IDF of each row of V^T is computed then all values in that row that are less than or equal to the average are set to zero. The goal of this pre-processing step is to strengthen the signal of the core sentences related to a concept while reducing the overall effect of other sentences. [16] After zeroing the qualified entries of V^T , the column sums, called the sentence *lengths*, are computed and the columns with the greatest lengths are chosen.

2.3 The TextRank Algorithm

An extractive, graph-based summarization method, TextRank follows the same paradigm as LSA but is based on an indicator approach for the intermediate document representation. TextRank (in the form employed in this experiment) uses sentence embeddings comprised of word vectors to represent sentences and chooses the n most closely related sentences, i.e., those whose vector representations’ directions are closest in the vector space. In computing the closeness of sentence vectors, the method implicitly extracts the concept features in an unsupervised way. The selection step for the TextRank algorithm is accomplished through weighted graph analysis. The algorithm, introduced by Mihalcea and Tarau in [11], is actually based on Google’s PageRank algorithm.

2.3.1 Sentence Embeddings

Mapping words from a vocabulary to a continuous vector space is a mathematical process called *embedding*. These word embeddings, or distributional vectors, for words are represented by a n -dimensional vector where $n \ll V$, where V is the size of the vocabulary. They attain an advantage over representations such as *one-hot encoding* by requiring smaller storage and computational requirements, thus avoiding things such as the *curse of dimensionality*. [20] In NLP, generating these mappings is an unsupervised learning problem, essentially clustering words by their meaning or direction in the vector space. These word embeddings are typically generated using neural networks.

Sentence embeddings are vector representations of sentences. The embeddings used in this experiment were constructed through element-wise averaging of the distributional vectors representing the words in the sentences. The word embeddings used were trained using the *fastText* approach introduced by Mikolov et al. in [13] that is based on the *word2vec deep learning* model for construction. The basic overview of this approach is to predict the word’s embedding using its context embedding (i.e., embeddings of the surrounding words) as the input to a deep neural network. This is called the *skip-gram model*; see [12] for more details.

2.3.2 Sentence Similarity and Selection

There are various sentence similarity measures: string kernels, cosine similarity, longest common sub-sequence, and so on. Cosine similarity was used in the TextRank algorithm of this experiment. The cosine similarity between two sentence vectors X and Y is defined

in [7] by

$$\cos(X, Y) = \frac{\langle X, Y \rangle}{\|X\|_2 \cdot \|Y\|_2} = \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i (x_i)^2} \cdot \sqrt{\sum_i (y_i)^2}}. \quad (2)$$

Once the cosine similarity measures are computed, they are taken as the weights to construct a directed graph with the sentences themselves represented by the graph’s vertices. Google’s PageRank algorithm is applied to the graph to rank the sentence nodes; however, instead of the weights representing probabilities of web-page landings (the PageRank score), they represent the similarity measures of eq. (2).

According to Mihalcea and Tarau in [11], the TextRank model computes the importance, or score, of a vertex within a graph ”based on global information drawn recursively from the entire graph.” Let $G = (V, E)$ be a directed graph with set of vertices V and set of edges E , where $E \subset V \times V$. For a vertex V_i , let $In(V_i)$ be the set of vertices that point to it, and let $Out(V_i)$ be the set of vertices that V_i points to. Then the score of vertex V_i is defined by

$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j), \quad (3)$$

where d is a damping factor usually set to 0.85. [11] The top n sentences (vertices), ranked by their score given by eq. (3) in decreasing order, are selected to construct a summary.

2.4 Evaluation Measures of Text Summaries

The automatic summaries generated in this experiment were evaluated using the ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation-L) measure described in [9] and cosine similarities defined by eq. (2). In each test, the generated summary was compared to an ideal summary—the TED Talk’s description or a human-generated summary. According to Steinberger and Ježek in [7], evaluation measures for determining the quality of automatic summaries are divided into *intrinsic* (e.g., text quality and content) and *extrinsic* (e.g., document categorization, question answering, etc.) categories. *Co-selection* and *content-based* measures were the metrics focused on in this experiment. The former examine automatic summaries as an overall construction of ideal sentences; measures in this category include precision, recall, and F-score. Content-based metrics, such as cosine similarity of words or the ROUGE-N, evaluate based on the similarity of n -grams, comparing words rather than entire sentences. [7]

The ROUGE-L test is based on the longest common subsequence (LCS) between the generated X and ideal Y summaries, $LCS(X, Y)$. The main idea is that summaries sharing a longer LCS will be more similar in meaning. *Precision* (P_{lcs}) and *recall* (R_{lcs}) are defined

by

$$P_{lcs} = \frac{LCS(X, Y)}{m} \quad (4)$$

$$R_{lcs} = \frac{LCS(X, Y)}{n}, \quad (5)$$

where m and n are the number of words in the generated and ideal summaries, respectively. F -score (F_{lcs}) is computed as a composition of P and R:

$$F_{lcs} = \frac{(1 + \beta^2) \cdot P_{lcs} \cdot R_{lcs}}{\beta^2 \cdot P_{lcs} + R_{lcs}}, \quad (6)$$

where β is a factor that gives more weight to precision when $\beta > 1$ and to recall otherwise.

3 Algorithm Implementation and Development

3.1 Loading and Pre-Processing Transcript Data

Removing *stop words*—i.e., function words such as *the*, *is*, *and*, for example—and punctuation was completed in the `preprocess_transcripts()` function in `utils.py` of [18]. The `stop_words` global variable references a list imported from the `nltk.corpus` package. [14] Word *stemming* was an optional task in this function. The `PorterStemmer()` class of the NLTK Python package has a method `stem()` which reduces sentences their root words (e.g., *honeybee* is replaced by *honeybe* and *city* by *citi*). The NLTK `PorterStemmer()` class, as suggested by its name, uses the Porter stemming algorithm. (See the NLTK documentation at <https://www.nltk.org> for details.) Figure 1 shows a Python code snippet from `preprocess_transcripts()` that carries out these tasks.


```

1 ps = PorterStemmer()
2 for t in transcripts:
3     pt = pd.Series(t).str.replace(r"(\(|\)|\.|\,|'|\")|([^\s-])", ' ')
4     if stem_words:
5         pt = [' '.join([ps.stem(j.lower()) for j in w.split() \
6                         if j not in stop_words]) for w in pt]
7     else:
8         pt = [' '.join([j.lower() for j in w.split() \
9                         if j not in stop_words]) for w in pt]
10    clean_transcripts.append(list(filter(None, pt)))

```

Figure 1: This code snippet from the `preprocess_transcripts()` function in `utils.py` of [18] shows how punctuation marks, words in parentheses, and stop words are removed from every sentence in addition to every word being stemmed if the `stem_words` flag is set to `True`.

3.2 LSA Implementation

Figure 2 below shows the Python code of the `get_concepts_df()` function in `utils.py` from [18]. If the `tags` parameter is not an empty list or Python’s `None` built-in constraint, the function counts the occurrence of each tag in the pre-processed transcript and stores the concepts in a list in descending order that have a word count greater than zero (lines 4 through 11 of fig. 2). Otherwise, the function extracts the concepts based on word frequency (lines 12 through 14 of fig. 2).

The descending order of document frequencies in which the concepts were stored were taken as the order of importance in the document. This order of importance is a significant aspect to Gong and Liu’s method in [6] for selecting the summary sentences (see 2.2.2).

Figure 3 below show a code snippet from the `topic_representation()` function. This function computes the TF-IDF of each concept for every sentence in the document. It returns a 2-D Numpy array of the topic representation matrix of §2.2.1.

```

1 def get_concepts(transcript, summary_length, tags=None):
2     if tags is not None:
3         concepts_struct = {}
4         for w in tags:
5             concepts_struct[w] = sum(1 for _ in re.finditer(r'\b%s\b' \
6                 % re.escape(w), ' '.join(transcript)))
7         sorted_struct = {k:v for (k,v) in sorted(concepts_struct.items(),\
8             key=operator.itemgetter(1), reverse=True) if v ...
9             > 0}
10        concepts = [k for k,_ in sorted_struct.items()]
11    else:
12        word_frequencies = Counter(' ...
13            '.join(transcript).split()).most_common()
14        concepts = [word_frequencies[j][0] for j in range(summary_length)]
15    return concepts

```

Figure 2: The `get_concepts()` function takes the record *tags* and *transcript* entries to find which tags are contained in the transcript and their word counts in descending order, stored in a Python dict. The returned dict entries are taken as the concepts of the document (transcript).

```

1 total_docs = len(transcript)
2 X = np.zeros((len(concepts), total_docs))
3 for j in range(X.shape[0]):
4     sentences_with_concept = 0
5     for k in range(X.shape[1]):
6         sentence = transcript[k].split()
7         word_count = sentence.count(concepts[j])
8         if word_count > 0:
9             sentences_with_concept += 1
10            X[j,k] = word_count/len(sentence)
11        idf = math.log2(total_docs/sentences_with_concept)
12        X[j,:] * idf
13    return X

```

Figure 3: Python code snippet from the `get-topic-representation()` function of [18]. This function computes the TF-IDF of each concept for each sentence and stores them in the Numpy array `X`. The variable `transcripts` is a list of strings representing sentences, and the `concepts` variable is a list of strings as well.

Gong and Liu’s method is implemented in the `gong_liu()` function in `lsa.py` of [18]. This function finds the column indices corresponding to the sentences with the highest TF-IDF values in each row of the matrix V^T and returns them in a Python list. The Python implementation for the cross method outlined in §2.1.1 is shown in fig. 4 below.

```

1 def cross(Vh, transcript):
2     for j in range(Vh.shape[0]):
3         Vh[j,:] = np.absolute(Vh[j,:])
4         avg_score = np.average(Vh[j,:])
5         for jj in range(len(Vh[j,:])):
6             if Vh[j,jj] ≤ avg_score:
7                 Vh[j,jj] = 0
8
9     sentence_lengths = [np.sum(Vh[:,j]) for j in range(Vh.shape[0])]
10    summary_indices = []
11    for _ in range(Vh.shape[0]):
12        I, _ = max(enumerate(sentence_lengths), key=operator.itemgetter(1))
13        summary_indices.append(I)
14        sentence_lengths[I] = 0
15    return summary_indices

```

Figure 4: The `cross()` function implements the cross method for selecting sentences outlined in §2.1.1. This function returns the indices of the sentences stored in the `transcripts` Python list returned by the `preprocess_transcript()` function.

Places of improvement are combining all words into root words, using n-grams,

3.3 TextRank Algorithm Implementation

The main code to implement the TextRank algorithm is located in `textrank.py` of [18]. This file contains two functions, `get_sentence_similarities()` and `summarize_text()`. The former, shown in fig. 5 below, uses the `cosine_similarity()` function of the `scikit-learn` Python package [17] to compute the measures given by eq. (2). These values are stored in a 2-dimensional Numpy array reference by the variable `similarity_matrix`. The similarity matrix is converted to graph representation using the `nx.from_numpy_array()` method of the `NetworkX` Python package of [15]. Finally the `nx.pagerank()` method is called to apply the PageRank algorithm and rank the sentences (represented by the nodes of the graph) by strongest relationship (cosine similarity) among all sentences. The TextRank implementation was inspired by Joshi’s article [8].

```

1 def get_sentence_similarities(sentence_vectors):
2     l = len(sentence_vectors)
3     similarity_matrix = np.zeros((l,l))
4     for j in range(l):
5         for k in range(l):
6             if j != k:
7                 similarity_matrix[j][k] = cosine_similarity(\
8                     sentence_vectors[j].reshape((1,300)),\
9                     sentence_vectors[k].reshape((1,300)))[0,0]
10    nx_graph = nx.from_numpy_array(similarity_matrix)
11    return nx.pagerank(nx_graph)

```

Figure 5: Python code of the `get_sentence_similarities()` function. See the paragraph above for a description.

The word vectors were obtained by using a pre-trained embedding published by Facebook called fastText. [13] The `load_wordvectors()` function code was taken from [13]. The `summarize_text()` function of `textrank.py` of [18] shows how the ranked sentence embeddings computed in the `get_sentence_similarities()` function were referenced and used to generate the text summarization.

3.4 Automatic Summary Evaluation

The ROUGE-L and cosine similarity measures were computed for each summary and its ideal counterpart by the `Evaluation` class in `utils.py` of [18]. The following code in fig. 6 below has a code snippet showing the implementation of the tests in the `rouge_l` and `cos_similarity()` methods. The PyRouge package of [19] by Pengcheng Yin was used to compute the ROUGE-L measure. Both methods take Python strings `s` and `l` representing the generated and ideal summaries respectively. The parameter β of eq. (6) was set to 1, giving equal weight to both precision and recall defined by eq. (5) and eq. (5) respectively. The generated summaries were evaluated against the TED Talk descriptions given in the raw data and against human generated summaries for the first five records.

```

1 @staticmethod
2 def rouge_l(S, I):
3     r = Rouge()
4     [precision, recall, f_score] = r.rouge_l([S], [I])
5     return precision, recall, f_score
6
7 @staticmethod
8 def cos_similarity(S, I):
9     summary_vecs = embed_sentences([w for w in S.split(' ')])[0]
10    ideal_vecs = embed_sentences([w for w in I.split(' ')])[0]
11    v1 = np.average(summary_vecs, axis=0)
12    v2 = np.average(ideal_vecs, axis=0)
13    return cosine_similarity(v1.reshape((1,300)),v2.reshape((1,300)))[0,0]

```

Figure 6: Python code of the `get_sentence_similarities()` function. See the paragraph above for a description.

4 Experiment Results

4.1 Evaluating Concept Extraction in LSA-Based Methods

Figure 7 below shows the *word cloud* for the TED Talk transcript, "Every City Needs Healthy Honey Bees." Stop words were removed from the transcript, and only words mentioned three or more times are shown in the visualization. This graph along with the singular value spectrum of fig. 8 were useful for in evaluating the performance of the method used for extracting the most important document concepts. It was observed from fig. 7 that the majority of high-frequency words were not included in the tags of this TED Talk. (The tags presumably are meant to be descriptors of the talk used for search-engine optimization, document classification, etc.) Words such as *honeybees*, *urban*, and *think* were not included in the tags and thus were not selected by the method of the `get_concepts()` function as being important concepts. This exclusion negatively impacted the quality of the generated summaries compared to the ideal summary (i.e., the human generated summary in table 1 below). Therefore, based on the LSA-based methods of this report, the tags selected for this transcript did not adequately represent the most important concepts.

The singular value spectrum in fig. 8 is essentially another interpretation of fig. 7, as it showed the "importance" of each concept in the topic representation matrix. The concepts extracted were, in decreasing order of word count, *bees*, *cities*, *agriculture*, *insects*, *plants*, and *science*. (Only six of the nineteen tags were actually mentioned in the talk.) Note that the word counts for the first three concepts, *bees*, *cities*, and *agriculture*, were 29, 7 and 4, respectively; the latter three were only mentioned once each. The graph in fig. 8

he TED Talk description was the following:

Bees have been rapidly and mysteriously disappearing from rural areas with grave implications for agriculture. But bees seem to flourish in urban environments—and cities need their help, too. Noah Wilson-Rich suggests that urban beekeeping might play a role in revitalizing both a city and a species.

Given this description, it would be nearly impossible to glean the description’s information from the automatic summary. However, this level of performance is on par with most extractive summarization methods that are not based on deep learning models, which are sometimes trained on gigabytes of data. The trade-off between computational costs and summary quality can be mitigated by combining deep learning models and those discussed in this report.

Table 1 below shows three-sentence summaries generated by a human reader, Gong and Liu’s and the cross methods for LSA, and the TextRank approach. Perhaps the most important sentence that captured the main ideas of the talk was, “What can you do to save the bees or to help them or to think of sustainable cities in the future?” It contained the top two most prominent words *bees* and *cities*, yet none of the methods selected this sentence. For the LSA-based methods, this was due to the independence among concepts in the selection process. That is, one maximum-value sentence was chosen per concept. Moreover, Python’s `max()` function chooses the maximum in the array with the lowest index if there are equivalent values present. To work around the latter issue, the transcript sentences could have been shuffled before applying Gong and Liu’s method. The cross method would not be affected by this, but it was affected by the concept-independent nature of the selection process.

Table 1: Human-generated and automatic summaries of the TED Talk, "Every City Needs Health Honey Bees."

Method	Three-Sentence Summaries
Human-Generated	<i>So honeybees are important for their role in the economy as well as in agriculture. We need bees for the future of our cities and urban living. What can you do to save the bees or to help them or to think of sustainable cities in the future?</i>
LSA-Gong&Liu	<i>This man is wearing what we call a bee beard. We think, "Oh, bees, countryside, agriculture," but that's not what the bees are showing. The bees like it in the city.</i>
LSA-Cross	<i>A beard full of bees now this is what many of you might picture when you think about honeybees maybe insects or maybe anything that has more legs than two. This man is wearing what we call a bee beard. This man is wearing what we call a bee beard.</i>
TextRank	<i>And let me start by telling you, "I gotcha." What's it gonna look like. This man is wearing what we call a bee beard.</i>

The TextRank algorithm, since it does not have the concepts explicitly defined, unsurprisingly chose sentences that were largely different semantically. The TextRank-generated summary in table 1 indicated that method was not influenced by the removal of stop words in the pre-processing step. This was deduced from the fact that the sentences of the TextRank summary were semantically related or comprised of prominent features (i.e., words or concepts) of the document text—e.g., *we*, *I*, and *bees*. The first two concepts were removed as stopwords, but the fastText word embeddings that yielded the sentence vectors were trained by observing the context surrounding words (see §2.3.1, [13], [12] for discussions around the skip-gram model). Therefore, the TextRank algorithm did not process the data based on the same set of features; hence the large semantic difference between the TextRank and LSA results.

Table 2 below shows the evaluation results produced with doing a permutation on of the order of significance of the concepts and word stemming on two sets of tests. Although the singular value spectrum of fig. 8 and the graph of fig. 7 portrayed more extreme prominence in the concepts, the data in table 2 indicated that the prominence of the main four concepts were not significant towards the summaries regarding these evaluation metrics. The data in table 2 did indicate a slightly better performance by the cross method over Gong and Liu on this particular document. The data also indicated that word stemming led to slightly better F-scores for both methods and improvement in the cosine similarity metric for Gong and Liu only. (Stemming was not applicable to the TextRank algorithm because the fastText word

embeddings in this experiment were not trained on stemmed text.)

Table 2: Evaluation Scores for LSA and TextRank Summaries for TED Talk "Every City Needs Healthy Honeybees." The prominence of concepts was permuted for each test set, and word stemming was performed for two sets of tests.

Method	Concepts	ROUGE-L F-Score	Cosine Similarity
LSA-Gong&Liu	[bee, cities, agriculture]	0.333	0.324
	[bee, agriculture, cities]	0.330	0.412
	[agriculture, bee, cities]	0.341	0.422
	[cities, agriculture, bee]	0.355	0.412
	Average	0.332	0.392
LSA-Gong&Liu (Stemmed Words)	[bee, cities, agriculture]	0.352	0.422
	[bee, agriculture, cities]	0.341	0.422
	[agriculture, bee, cities]	0.361	0.438
	[cities, agriculture, bee]	0.355	0.412
	Average	0.352	0.423
LSA-Cross	[bee, cities, agriculture]	0.406	0.462
	[bee, agriculture, cities]	0.446	0.461
	[agriculture, bee, cities]	0.446	0.461
	[cities, agriculture, bee]	0.446	0.461
	Average	0.436	0.461
LSA-Cross (Stemmed Words)	[bee, cities, agriculture]	0.446	0.461
	[bee, agriculture, cities]	0.446	0.461
	[agriculture, bee, cities]	0.446	0.461
	[cities, agriculture, bee]	0.446	0.461
	Average	0.446	0.461

4.2 Document and Summary Length Effects

The data in table 3 below showed that Gong and Liu’s LSA method performed better in every evaluation metric on average for the first five TED Talk transcripts. One of the two LSA-based methods outperformed the TextRank method in every evaluation measure. This was due to multiple factors: the fastText word embeddings and how they were trained on a different data set (Wikipedia pages), the length of the transcripts being summarized, and the length of the target summary, among others. For example, the cosine similarity measure tended to be higher for the methods applied to the TED Talk "Simplicity Sells," the longest of the first five transcripts. The bar plot in fig. 9 below shows the word counts for the first five transcripts. From this plot and the table 3 data, another trend observed in how the LSA-based methods achieved higher F-scores on the shorter transcripts.

Table 3: Method Performance with Human-Generated Ideal Summaries.

Method	Transcript	ROUGE-L			Cosine Similarity
		Precision	Recall	F-Score	
LSA-Gong&Liu	0	0.447	0.364	0.401	0.893
	1	0.810	0.193	0.312	0.268
	2	0.769	0.193	0.309	0.361
	3	0.585	0.386	0.465	0.484
	4	0.600	0.801	0.686	0.588
	Average	0.642	0.387	0.435	0.519
LSA-Cross	0	0.417	0.286	0.339	0.731
	1	0.592	0.276	0.377	0.262
	2	0.438	0.411	0.424	0.332
	3	0.415	0.494	0.451	0.425
	4	0.413	0.458	0.435	0.367
	Average	0.455	0.385	0.405	0.423
TextRank	0	0.330	0.471	0.388	0.655
	1	0.528	0.376	0.439	0.356
	2	0.458	0.444	0.451	0.358
	3	0.548	0.261	0.354	0.512
	4	0.572	0.315	0.406	0.660
	Average	0.487	0.374	0.408	0.508

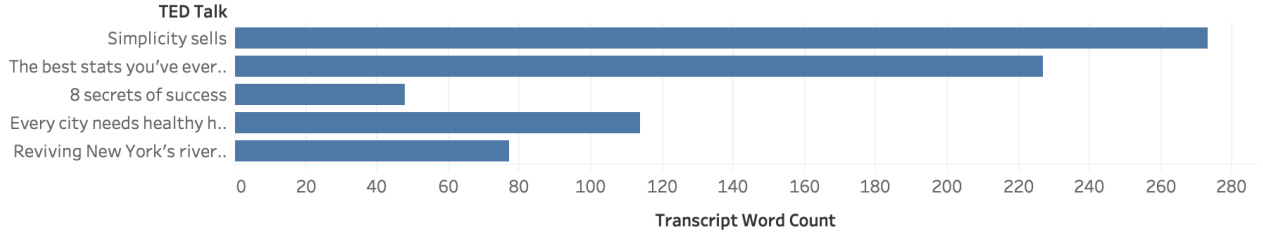


Figure 9: Bar plot showing the total word count in the first five transcripts of the data set. The data in table 3 above shows trends in how the cosine similarity measures were higher for the methods applied to the TED Talk "Simplicity Sells," the longest of the first five transcripts, and the the LSA-based methods achieved higher F-scores for the shorter transcripts.

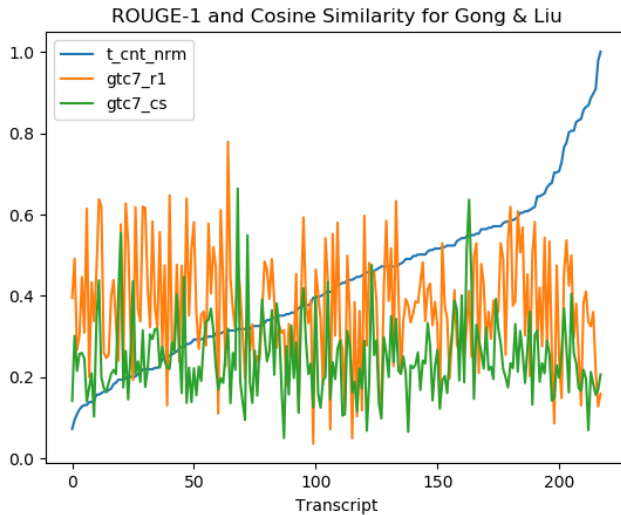


Figure 10: ROUGE-1 F-Score and cosine similarity measures for Gong and Liu’s selection method against normalized transcript word counts. As the length of the transcript increases, there is a slight reduction in the evaluation scores.

Table 4 below shows the average precision, recall, F-score, and cosine similarity over 50 transcript summaries generated by the LSA and TextRank methods. The LSA-based method of Gong and Liu achieved the highest F-score and cosine similarity scores of all methods. The LSA-based methods performed on par with the methods given in the literature [16] and [6]. Currently there are no apparent published works on text summarization using the TED Talk data set of this report; however, there are encompassing claims in the existing literature, such as those by Das and Martins in [4] and Wang et al. in [3], that efficiently generating summaries of short length, like those of this report, is a difficult task for extractive methods, and finding a small set of sentences to convey the main ideas of a document is also difficult.

Table 4: Evaluation Scores for LSA and TextRank Summaries Averaged Over 50 Documents with summary lengths ranging from 2 to five—the length of the corresponding talk descriptions.

Method	ROUGE-L			Cosine Similarity
	Precision	Recall	F-Score	
LSA-Gong&Liu	0.355	0.515	0.367	0.592
LSA-Cross	0.310	0.561	0.333	0.487
TextRank	0.325	0.528	0.364	0.563

Table 5: Evaluation Scores for LSA and TextRank Summaries Averaged Over 218 Documents where the generated summaries were all of length seven, i.e., consisted of seven sentences.

Method	F-Score		Cosine Similarity
	ROUGE-1	ROUGE-2	
LSA-Gong&Liu	0.373	0.015	0.247
LSA-Gong&Liu No Tags	0.374	0.014	0.248
LSA-Cross	0.406	0.039	0.246
LSA-Cross No Tags	0.484	0.038	0.247
TextRank	0.325	0.528	0.563

Based on the results in table 5, the cross method for sentence selection produced better ROUGE-1 F-Scores than Gong and Liu’s method. The cosine similarity measures were virtually the same. In light of these results and those from §4.1, the transcript length had no observable effect on the ROUGE-N quality of the generated summaries.

The generated summaries of the block below each had lengths of 7. Arguably the TextRank-generated summary captured the most important points of the talk—i.e., the benefits of beekeeping in the city and the speaker’s adjuring to take part.

1. Cross method using tag concepts, *bees, cities, agriculture, climate change, insects, plants, science*:

A beard full of bees now this is what many of you might picture when you think about honeybees maybe insects or maybe anything that has more legs than two. He probably has a queen bee tied to his chin and the other bees are attracted to it. This man is wearing what we call a bee beard. This man is wearing what we call a bee beard. This man is wearing what we call a bee beard. This man is wearing what we call a bee beard. This man is wearing what we call a bee beard.

2. Cross method using high-frequency word concepts, *bees, I, honeybees, years, we, urban, think*:

I understand that. So this really demonstrates our relationship with honeybees and that goes deep back for thousands of years. A beard full of bees now this is what many of you might picture when you think about honeybees maybe insects or maybe anything that has more legs than two. and let me start by telling you, "I gotcha." But there are many things to know and I want you to open your minds here keep them open and change your perspective about honeybees notice that this man is not getting stung. He probably has a queen bee tied to his chin and the other bees are attracted to it. This man is wearing what we call a bee beard.

3. Gong and Liu’s method using tag concepts, *bees, cities, agriculture, climate change, insects, plants, science*:

This man is wearing what we call a bee beard. We think, "Oh bees, countryside, agriculture," but that's not what the bees are showing. The bees like it in the city. We have tar paper on the rooftops that bounces heat back into the atmosphere contributing to global climate change no doubt. A beard full of bees now this is what many of you might picture when you think about honeybees maybe insects or maybe anything that has more legs than two. Now this was a mystery in the New York Times where the honey was very red and the New York State Forensics Department came in and they actually did some science to match the red dye with that found in a maraschino cherry factory down the street. So we know flowers we know fruits and vegetables even some alfalfa in hay that the livestock for the meats that we eat rely on pollinators, but you've got male and female parts to a plant here and basically pollinators are attracted to plants for their nectar and in the process a bee will visit some flowers and pick up some pollen or that male kind of sperm counterpart along the way and then travel to different flowers and eventually an apple in this case will be produced.

4. Gong and Liu's method using high-frequency word concepts, bees, I, honeybees, years, we, urban, think:

This man is wearing what we call a bee beard. We think, "Oh bees, countryside, agriculture," but that's not what the bees are showing. The bees like it in the city. Think of the kids today. We don't even find dead bodies. Think of the kids today. The urban honey is delicious.

5. TextRank method:

It's just because people are uncomfortable with the idea, and that's why I want you today to try to think about this think about the benefits of bees in cities and why they really are a terrific thing. Let me give you a brief rundown on how pollination works. The bees in Boston on the rooftop of the seaport hotel where we have hundreds of thousands of bees flying overheard right now that I'm sure none of you noticed when we walked by are going to all of the local community gardens and making delicious healthy honey that just tastes like the flowers in our city. We need bees for the future of our cities and urban living. Here's some data that we collected through our company with best bees where we deliver install and manage honeybee hives for anybody who wants them in the city in the countryside, and we introduce honeybees and the idea of beekeeping in your own backyard or rooftop or fire escape for even that matter and seeing how simple it is and how possible it is there's a counter-intuitive trend that we noticed in these numbers. Honey is a great nutritional substitute for regular sugar because there are different types of sugars in there. We also have a classroom hives project where this is a nonprofit venture we're spreading the word around the world for how honeybee hives can be taken into the classroom or into the museum setting behind glass and used as an educational tool.

5 Summary and Conclusions

Data-driven analyses can provide great leverage for processing and understanding the vast amount of textual data that exists today, and the extractive text summarization methods explored in this report showed promise in this objective. Automatic text summarization is one tool of NLP with very active areas of research; these areas include state-of-the-art deep learning methods, novel and improved evaluation metrics, and so on.

Based on the ROUGE-N, ROUGE-L and cosine similarity evaluation measures, the LSA-based methods and TextRank produced summaries of competitive quality with the LSA methods performing slightly better. On the document examined in §4.1 in particular, however, the LSA-based methods produced summaries that were closer semantically to the main ideas of the document. This was caused by the differing natures of the approaches: the TextRank approach was essentially processing the data based on a different set of features (concepts) from the LSA methods. Although categorized as an unsupervised method, one LSA scheme of this report was supervised in that the concepts were derived based on data external to the document—the TED Talk tags. The TextRank features were derived through word embeddings trained on the context surrounding concept words instead of the concept words themselves. This explained the selection of sentences not containing the important concepts (see §4.1). Interestingly, the LSA-based methods produced similar scores whether the tags or the high-frequency words were taken as the document concepts.

For the TextRank method, targeting longer summaries seemed to yield the best summary, qualitatively, in the transcript case study for "Every City Needs Healthy Honeybees." The cross method produced less coherent summaries of this document as the target summary length increased, but Gong and Liu's method improved, in accordance with what the literature suggested. One possible area of improvement for TextRank would be the use of state-of-the-art contextualized word embeddings. These might have yielded better results since they would be a sort of refinement on the feature space. Additionally, it is common practice to train word embeddings on the actual data they are used for, so instead of using distributional vectors pre-trained on a different data set, word embeddings trained on a large TED Talk data set would have likely been better for realizing the feature space.

Various aspects were examined to alter the performance of the LSA-based methods such as removing stop words, stemming, and permuting concepts. While concept permutation had little affect, removing stop words improved the methods overall, and stemming yielded an increase in some metrics for the LSA-cross method. Generating longer summaries may have yielded better results for the LSA methods according to [4] and [3], but this was not an option in this experiment, where the ideal reference summaries were on average three

sentences long. As discussed in §4.1, the Python implementation of Gong and Liu’s method might have been improved by shuffling the sentences of the transcript.

Due to the naivety of the concept selection process of LSA (i.e., concepts having no influence on each other) a sentence that would be highly relevant based on TF-IDFs to more than one concept might be passed over in the selection process; and another sentence will be chosen if its TF-IDF is higher in the concept under consideration. That is to say, a highly important sentence to the overall document could be missed due to the nature of the LSA selection methods.

References

- [1] C. Aggarwal and C. Zhai, "Mining Text Data," Kluwer (2011).
- [2] R. Banik, *TED Talks Data Set*, available at <https://www.kaggle.com/rounakbanik/ted-talks>.
- [3] J. Carthy, J. Dunnion, and R. Wang, "Machine learning approach to augmenting news head-line generation," Proceedings of the International Joint Conference on NLP (2005).
- [4] D. Das and A.F.T. Martins, "A Survey on Automatic Text Summarization," (2007) available at <http://www.cs.cmu.edu/~nasmith/LS2/das-martins.07.pdf>.
- [5] G. Galliers and K. Jones, "Evaluating Natural Language Processing Systems: An Analysis and Review," Lecture Notes in Artificial Intelligence No. 1083, Springer (1995).
- [6] Y. Gong and X. Liu, "Generic Text Summarization using Relevance Measure and Latent Semantic Analysis," Proceedings of 24th International ACM SIGIR Conference (2001).
- [7] K. Ježek and J. Steinberger, "Evaluation Measures for Text Summarization," Journal of Computing and Informatics **28** (2009).
- [8] P. Joshi, *An Introduction to Text Summarization using the TextRank Algorithm (with Python implementation)*, available at <https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/>.
- [9] C. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," Journal of Computing and Informatics **28** (2009).
- [10] H.P. Luhn, The Automatic Creation of Literature Abstracts, IBM Journal of Research and Development, **2**, 2 (1958) pp. 159-165.
- [11] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," Proceedings of the 2004 Conference on Empirical Methods in NLP (2004) pp. 404-411.
- [12] T. Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality," Advances in Neural Information Processing Systems **26** (2013).
- [13] T. Mikolov et al. "Advances in Pre-Training Distributed Word Representations," Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).

- [14] *Natural Language Toolkit, NLTK 3.4 Documentation*, available at <https://www.nltk.org>.
- [15] *NetworkX*, available at <https://networkx.github.io/>.
- [16] M.G. Ozsoy et al., "Text Summarization using Latent Semantic Analysis," *Journal of Information Science* **37**, 4 (2011).
- [17] F. Pedregosa et al., Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, **12** (2011) pp. 2825-2830.
- [18] J. Sellers, *Extractive Text Summarization GitHub Repository*, available at <https://www.github.com/johnsell620/extractive-text-summarization>.
- [19] P. Yin, *PyRouge*, available at <https://github.com/pcyin/PyRouge>.
- [20] T. Young et al., "Recent Trends in Deep Learning Based Natural Language Processing," arXiv:1708.02709.