Abstractive Multi-document Summarization using Topical Simplicial Curves

João Cruz
Instituto Superior Técnico, Universidade de Lisboa
joao.r.cruz@tecnico.pt

Abstract

We explore the effectiveness of simplicial curves, a word-representation method that is context-sensitive, motivated by its intrinsic mathematical properties (e.g., differentiation and ease of combining representations), in the multi-document summarization task. We use the DUC 2006 and DUC 2007 multi-document summarization corpora. The generated summaries are compared with the reference summaries using the ROUGE-1, ROUGE-2 and ROUGE-L evaluation metrics. Compared to the ROUGE-1 score of 0.29 of the simplest chosen baseline, our method achieves a ROUGE-1 score of 0.04, falling below our expectations. We conclude with an exploration of the obtained results and suggest other applications of the simplicial curves method.

Keywords: Multi-document Summarization, Simplicial Curves, Word Representations

1. Introduction

Automatic text summarization is a task where the goal is to, given one or more documents, produce a small text that accurately captures the information contained in the documents being summarized. This can be mainly done in two ways: a) by selecting important words or passages of the original text to preserve in the summary — extractive summarization, or b) by generating new words (or new sentences with the original words) that better synthesize what was in the original text (i.e., rephrase the text) — abstractive summarization. Despite working towards the same goal, these two approaches have differences in metrics and corpora used, and implementation methods.

For this purpose, is it fundamental to have a foundational framework that transforms the words of a text (such as the ones in this sentence) into objects that can be manipulated using mathematical operations, and, as such, can be used in computer applications. Different transformation methods encode different aspects of language (such as syntactic properties), and using one over the other is usually a matter of what trade-off is acceptable for a particular application. For example, if we are interested in automatically tagging the parts-of-speech of a text, we should choose a representation that enhances its syntactical aspects.

In this work, we are concerned with the re-exploration of a method to represent text in a manner that deviates from the current, well-established representation methods. This method — simplicial curves (Lebanon et al., 2007) — was chosen for its rich mathematical properties and the potential for finding parallels between fundamental analytical operations (integrals, derivatives) and results in the textual domain. Also, simplicial curves inherently encode the sequencing of text and, since we can define an algebra over this representation, lend themselves to composition. Simplicial curves have an intuitive sense of a document traversing in the space composed by its parts (often words but, as we will see, we can admit other definitions for lexical units), which can help with the explainability of the obtained results.

Our task is to explore the effectiveness of the simplicial curves approach in the field of abstractive multi-document summarization (MDS — produce a textual summary from multiple documents, possibly from different sources, and talking about the same thing, while focusing on slightly different aspects), as opposed to single-document summarization (SDS — produce a textual summary from a single document), using MDS datasets. We also explore different methods of combining documents in a single representation and extracting text from it, evaluating our results using standard summarization literature metrics (ROUGE (Lin, 2004); see section 6).

This document is structured as follows: a) section 2 introduces historical context for different word representations in linguistic tasks; b) section 3 introduces the notion of simplex and objects embedded in it; c) section 4 builds an algebra of curves, showing how we can combine two curves into a third one or transform a curve into a numeric value; d) section 5 introduces the most used corpora in MDS; e) section 6 introduces the most used evaluation metrics in MDS; f) section 7 addresses the related work done in the MDS task, both fundamental and state-of-the-art; g) section 8 delimits the corpora, evaluation metrics and baselines that we have used in the MDS task; h) section 9 presents the results from our experiments; and i) section

10 concludes the document, also pointing out some future work to be done.

2. Background on Word Representations

In the following sections we will provide a description of different dimensional word representations. We are interested in different types of word representations because they will serve as the base representation upon which the simplex will be built.

2.1. Traditional Representations

The most basic transformation from words to a mathematical object that can be manipulated is called the one-hot approach. In this approach, words correspond to dimensions in some space $\mathbb{N}^n$, and a word vector is represented by a 1 in the position for the word and 0 everywhere else. For example, if we have a text with two words, "test" and "red", the vector for "test" is $(1, 0)$ and for "red" is $(0, 1)$.

This basic model was further improved with the use of term-frequency (TF) (Luhn, 1957), where a word is represented by a one-hot vector multiplied by the word's frequency in some document, which then is composed of stacks of word-vectors (a matrix). Another possible definition is that a document is the sum of the vectors of the words that are in the document. Under this model, a word is considered important in a document if it appears many times in it. This approach does not perform well in downstream applications when applied to texts where something as simple as function words (such as "the" — the most frequent word in English) are abundant, which are most texts. To counter this, the TF approach was enhanced by the addition of an "inverse-document-frequency" (IDF) term (Jones, 1972), where the multiplicative component of TF is weighted down by a function of the number of documents in which the word occurs. The combination of both methods is known as TF-IDF:

$$\text{tfidf}(w, d) := \text{tf}(w, d) \times \text{idf}(w) \tag{1}$$

$$\text{tf}(w, d) := \frac{\text{times } w \text{ appears in } d}{\max\{\text{times } t \text{ appears in } d \,|\, \forall t \in d\}} \tag{2}$$

$$\text{idf}(w) := \log \frac{\#D}{\#\{d \in D \,|\, w \in d\}} \tag{3}$$

where D is a corpus, $d \in D$ is a document, and $w, t \in d$ are word in that document. Note that there are other sensible definitions for the TF function; we chose to define it as the normalized frequency.

Representing words as TF-IDF vectors accurately modeled text for many applications, but this method for word representation results in vectors that are too sparse for applications such as text classification and others. To counteract this, Latent Semantic Indexing (Deerwester et al., 1990) took the resulting sparse matrix and decomposed it using Singular Value Decomposition, allowing for the extraction of dense representations for words and documents that allowed the use of other similarity notions, e.g., two words/documents are similar if the cosine of the angle between their vectors is close to 1.

2.2. Neural Representations

Although representations derived from neural networks were already being studied (Collobert and Weston, 2008; Turian et al., 2010; Mnih and Hinton, 2007) since 2000 with their introduction to NLP in Bengio et al. (2000), it was in 2013 that the revolution of dense vector space representations of words derived from a neural network was kick-started by Mikolov et al. (2013) (here called SGNS, after the main method: "skip-gram with negative-sampling"). In this approach, word vectors are extracted from the inner state of a neural network after training on some proxy task (in their case, word similarity of a random word with the rest of the words in the enclosing sentence). The advantages of SGNS-based representations are that a) the resulting word vectors capture A:X::B:Y (A is to X as B is to Y) analogies by means of simple arithmetic: if we want to solve A:X::B:? using SGNS embeddings, we can find the vector of ? by $v_? = v_X - v_A + v_B$; b) they are easy to incorporate in downstream applications, seeing as to get a word-embedding for a word w, all one has to do is lookup the row of w in the embedding matrix W (serving as a lookup table), after the neural network has been trained; and c) they indiscriminately improved task scores on many different NLP benchmarks (Baroni et al., 2014).

Subsequent research expanded on this trend, with various extensions and modifications appearing over the decade. Of note are fastText in 2017 (Bojanowski et al., 2017) (rather than considering words to be the smallest textual unit, instead build vectors for each character in the text; a word vector is then the sum of the vectors of its characters), ELMo in 2018 (Peters et al., 2018) (words have different vector representations, depending on the context in which they appear), and more recently BERT in 2019 (Devlin et al., 2019) (a generalization of ELMo).

In 2014, a count-based approach (like TF-IDF) called GloVe was presented by Pennington et al. (2014), which aimed to compete with SGNS-embeddings in their representational power. Although their results in the word analogy, word similarity and named entity recognition tasks indeed showed better results than SGNS-based solutions, Levy et al. (2015) later showed empirically that GloVe is worse than SGNS for the word analogy and word similarity tasks on various datasets (they do not conclude that representations derived from counting approaches are worse than those derived from neural approaches, however), which reveals the variance of the impact of the representation for a given task, illustrating the importance of choosing the appropriate word representation for the task at hand. Moreover, further research (Levy and Goldberg, 2014) showed that SGNS are doing nothing more than factoring a Pointwise Mutual Information matrix derived from the text. In fact, how effective a method is in solving A:X::B:? analogies was discovered by Ethayarajh et al. (2019) to be caused by

variance shifts in a modified formulation of the Pointwise Mutual Information between any two words.

Also, one major drawback of any of the above representations (apart from ELMo and BERT, indirectly) is that word order in a text is not preserved in the vector representation. These so-called bag-of-words methods construct their vectors as if any two words in a text could be inter-exchanged, which is a fundamental oversight: text is sequentially structured, and we can obtain much information by order alone (e.g., in SOV languages we know that the last word in a sentence is most likely the verb). The relevance of this order information is, nevertheless, dependent on the task.

3. Objects in the Simplex

In the following sections we describe the notion of a simplex over an object space, and of curves in this simplex.

3.1. Vocabulary Simplex

A simplex $\mathbf{\Sigma}_n$ over some set C of size n is a subset of an $\mathbb{R}^n$-dimensional space where each dimension represents an object in C, and the coordinates of each point in this subset are non-negative and sum to one, i.e., $\mathbf{\Sigma}_n := \{\mathbf{x} \in \mathbb{R}^n \,|\, x_i \geq 0, \sum_{i=1}^{n} x_i = 1\}$. We can thus understand points in the simplex as probability distributions over the objects in C. We call C the collection and its objects the items. These can be concrete (such as real words in a vocabulary) or abstract (such as the topics of a document).

Geometrically, the simplex can be thought of as an $(n-1)$-dimensional triangle, e.g., $\mathbf{\Sigma}_3$ is the surface $x+y+z=1$. Each dimension is thus a vertex in this triangle, and the notion of probability in this space is how close (under the L_2 metric) a given point is to a vertex (see Fig. 1).

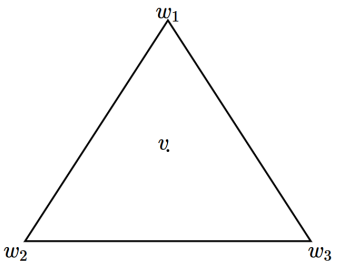

Figure 1: An example of Σ_3, where each vertex corresponds to a word. The middle point v is a distribution over each of the words. Being equidistant from all vertices, v is a uniform distribution, that is, $v = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

3.2. Curve Overview

Simplicial curves were first introduced by Lebanon et al. (2007), motivated by the, at the time, lack of representation methods to model sequential content in textual documents. As a generalization of the bag-of-words representation, simplicial curves aim to preserve the same vector-space analogy as traditional representation methods, with the added benefit of modeling words locally — i.e., the document is a time-dependent histogram of words rather than a plain vector of all the words in the document.

The main idea is to model a text document y (or any sequence of discrete objects) as a continuous, sequential mathematical object, i.e., a parametric curve, where one can use standard calculus tools (e.g. derivatives, integrals, metrics) to model properties of the sequence. This also introduces the concept of time (represented by μ) in a document, taken to be between 0 and 1. $\mu = 0$ represents the beginning of the curve, which maps to the beginning of the sequence, and $\mu = 1$ represents the end of the curve, mapping to the end of the sequence.

The way the curve is built is flexible in that it allows one to model the original sequence at different levels of detail. If we choose a lower level of detail then the curve will focus more on the individual objects of the sequence (in the case of text, words); if we choose a higher level of detail then the curve will tend to model the sequence as a whole. Different representations of the same curve (or even of different curves) can be combined to produce a single curve that models the sequences at a varying level of detail.

A curve can be seen as a function γ_y from $\mu \in [0, 1]$ (the point of the curve we want to query) to a member of $\mathbf{\Sigma}_n \subset \mathbb{R}^n$, a point in the simplex (i.e., a distribution) over the objects in the original space.

Intuitively, the curve is a mapping from the normalized position of the document to a histogram of the words in that position in the document. All curves are the same length to allow comparisons between curves built from different-length sequences.

3.3. Curve Construction

Given an item sequence y of length N, the method starts by building an initial $N \times \#C$ matrix, M_y, where the columns are the features of the objects (the vocabulary space) and the rows are the vector representations of each of them. The matrix indices are then made continuous in time ($t \in [0, 1]$) by making M_y be indexed not by its row numbers, but by a function φ defined as

$$\varphi_{M_y}(t, w) := [M_y]_{\lceil tN \rceil, w} \tag{4}$$

where $[M]_{i,j}$ indicates the (i, j)'th element of the matrix. M_y can be built in a variety of manners, which allows us to leverage several base-representations and the added benefits they contain. More details will be shown in the following sections.

Once the continuous-access matrix representation is obtained, we smooth the entire matrix by multiplying in time (so only the t variable is involved) the access function φ_{M_y} with some smoothing kernel $K_{\mu,\sigma}$, where $\mu \in [0, 1]$ is the center and $\sigma > 0$ is the scale of the kernel.

For simplicity, a convenient choice for a smoothing kernel is a restricted Gaussian in $[0, 1]$.

The scale parameter controls the shape of the kernel, which is fundamental to control the level of detail we want the curve to encode. If the kernel is too wide (i.e., the smoothing is too strong) then the multiplication will yield a not-too-detailed view of the document (this is, a higher-level representation of the document as a whole). If the kernel is too narrow (i.e., the smoothing is too weak) then the multiplication will yield a very peaky view of the document, precisely focusing on the original words. Formally:

$$\gamma_y^\sigma(\mu)_w := \int_0^1 \varphi_{M_y}(t, w) \times K_{\mu,\sigma}(t)\,\mathrm{d}t \tag{5}$$

where γ_y^σ is a distribution over the original collection C built using the items in y and $\gamma_y^\sigma(\mu)_w$ is the probability of item w from C at time $\mu \in [0, 1]$ in the sequence. For convenience, we will generally drop the y and σ indices, only using them when we need to refer to curves built from different documents or using different kernel scales.

One way of obtaining M_y is via the bag-of-words, where M_y is built by stacking one-hot vector representations of the words in the order that they appear in the sequence. To avoid the sparse representation pitfall (see section 2.1), we apply some form of smoothing. Any classical form of smoothing can be used so, for simplicity, we use Laplace Smoothing by $c \in \mathbb{R}$.

Note that the larger c is, the closer all the previously-zero indices of the one-hot vectors will be, which means that the corresponding point will tend towards the center of the vocabulary simplex. Since all words will be smoothed the same way, the resulting curve will predominantly stay near the center of the simplex.

This approach offers two advantages: (a) it is easier to grasp its intuition since every word is its own dimension, and (b) it is easy to implement. However, preliminary experimental results show that the resulting representation may not capture important information that should be in the summary. A possible explanation for this is that, since a one-hot vector space has no intrinsic meaning, we are not leveraging information contained in the vector spaces of other forms of base representations (LDA (Latent Dirichlet Allocation (Blei et al., 2003)) with topics, SGNS with semantic similarity).

More generally, M_y can be built using pre-built vector representations such as those given by methods like word2Vec (SGNS) (Mikolov et al., 2013) or ELMo (Peters et al., 2018). In this way, we can adapt the benefits that these representations provide (analogy resolution, dense base-representations) with what simplicial curves gives us (ordering, algebraic operations). The same normalization precautions described above also apply in this case.

4. Curve Algebra in the Simplex

The main motivating idea is doing importance selection in the curve space and then mapping the results back to text. To achieve this, we need to have a way of combining curves in various ways, emphasizing different points.

With that in mind, we can define a basic algebra in the simplex, using familiar concepts such as addition and concatenation, allowing us to combine different points (or sets of points), indirectly combining the original items.

Since a curve is just a function $\gamma : [0, 1] \to \mathbf{\Sigma}^n \subset \mathbb{R}^n$, we can consider the algebra of vector valued functions, with some additional operations to best allow the combination of curves, as long as they remain closed in the simplex.

Let γ_1 and γ_2 be two curves and γ' the result of combining both in some way. We define the following:

- Curve addition

$$\gamma'(\mu) = \frac{1}{2}\left(\gamma_1(\mu) + \gamma_2(\mu)\right) \tag{6}$$

- Curve subtraction

$$\gamma'(\mu) = \mathrm{softmax}\left(\gamma_1(\mu) - \gamma_2(\mu)\right) \tag{7}$$

- Curve concatenation

$$\gamma'(\mu) = \text{if } \mu < \frac{1}{2} \text{ then } \gamma_1(2\mu) \text{ else } \gamma_2(2\mu) \tag{8}$$

- Curve conflation

$$\gamma'(\mu) = \frac{\gamma_1(\mu) \otimes \gamma_2(\mu)}{\sum_w \gamma_1(\mu)_w \otimes \gamma_2(\mu)_w} \tag{9}$$

All of these generalize to a higher number of curves.

Curve addition has an intuitive motivation: just return the curve in the geometric space between γ_1 and γ_2.

Conflation (Hill, 2011) is a method used to compose different probability distributions over the same underlying objects whilst ensuring important statistical properties (e.g., conflation minimizes the loss of Shannon Information, and yields a maximum likelihood estimator, among others). The vector multiplications are done element-wise.

Curve subtraction runs the risk of yielding a negative probability distribution, so we need to normalize it to positive by applying the softmax function.

Concatenation also has an intuitive meaning — take the beginning of the second curve and attach it to the end of the first, correcting the access argument accordingly. This can be useful for, e.g., forming a curve for a document by sequentially composing the curves for its sentences.

Do note that it does not make sense to consider curve scaling by some scalar in $\mathbb{R}$ since we would immediately have to re-normalize, losing the scaling operation.

We can also define the curve inner-product, allowing us to see how much two given curves "agree" with each other, i.e., how similarly they travel in the simplex space.

- Curve inner-product

$$\gamma_1 \cdot \gamma_2 = \int_0^1 \gamma_1(\mu) \cdot \gamma_2(\mu)\,\mathrm{d}\mu \tag{10}$$

Since the result of evaluating a curve at a given point is a distribution, we can also generalize common probabilistic descriptors such as entropy or the Fisher information:

- Curve Entropy

$$\mathrm{H}(\gamma) := \int_0^1 \mathrm{H}(\gamma(\mu))\,\mathrm{d}\mu \tag{11}$$

- Curve Fisher Information

$$\mathcal{I}(\gamma^\sigma) := \int_0^1 \mathbb{E}_{w\sim\gamma^\sigma(\mu)}\left[\left(\frac{\partial}{\partial\sigma}\log(\gamma^\sigma(\mu)_w)\right)^2\middle|\sigma\right]\mathrm{d}\mu \tag{12}$$

We can also compare two different curves by finding their distance d under some metric $\mathcal{M}$, e.g., L_2 distance (yielding the mean euclidean distance of one curve to another in space), or the Jensen-Shannon metric, defined as $\mathrm{JS}(P\,\|\,Q) := \frac{1}{2}\left(\mathrm{KL}(P\,\|\,A) + \mathrm{KL}(Q\,\|\,A)\right)$, with $A = \frac{1}{2}\left(P+Q\right)$, where KL is the Kullback-Leibler divergence, yielding how similar two curves are from one another, in terms of their probability distributions.

- Curve difference under metric $\mathcal{M}$

$$d_{\mathcal{M}}(\gamma_1,\gamma_2) := \int_0^1 \mathcal{M}(\gamma_1(\mu),\gamma_2(\mu))\,\mathrm{d}\mu \tag{13}$$

5. Corpora

The Document Understanding Conference[1] (DUC) were a series of challenges running from 2001 to 2007 whose aim was to evolve the state-of-the-art in text summarization. To this end, a corpus for MDS was published each year, which invited competing implementations. The top-ranked systems became good baselines for MDS. Of note are the DUC 2006 and DUC 2007 corpora, which, for our purposes, are comprised of, respectively, 50 and 45 document clusters of English news from the Associated Press and the New York Times. Each document cluster has, on average, 25 documents.

Since 2008, DUC became the summarization track of the Text Analysis Conference[2] (TAC), where the goal was the same. The track ran from 2008 to 2011 (and uniquely in 2014) but, in recent years, TAC has grown to focus on knowledge-based systems. TAC challenges were more diverse, ranging from MDS to just summary evaluation, opinion summarization or even multilingual summarization. Of note is the TAC 2009 corpus, which is a dataset of 44 topics and 20 documents clusters per topic. The dataset is a subset of AQUAINT-2 (Vorhees and Graff, 2008), a collection of 907k documents in English, comprised from articles from October 2004 to March 2006 from Agence France-Presse, Central News Agency (Taiwan), Xinhua News Agency, Los Angeles Times, Washington Post News Service, New York Times and Associated Press.

Recently, Fabbri et al. (2019) introduced a new dataset for MDS along with some baselines on that dataset using MMR and end-to-end methods. It consists of 56216 documents taken from scrapped full news articles and summaries from `newser.com`. Each summary is handmade and has at least two or more sources from where it was obtained. The baseline methods used were also tested in DUC and reported worse performance than known values for DUC. This highlights the fact that the effectiveness of different summarization techniques is also highly dependent on the type of text we want to summarize.

6. Evaluation Metrics

The classical metric used to automatically measure summary quality is Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin, 2004), which measures lexical overlap between the produced summary and some reference summary in various ways. The overlaps can be computed at the word level (ROUGE-1), bi-gram level (ROUGE-2), bi-gram with n words in between (ROUGE-Sn), bi-gram with n words in between and uni-gram overlap (ROUGE-SUn), longest common subsequence (ROUGE-L), and some subsequent extensions considering dense vectors built from n-grams (ROUGE-n-WE) (Ng and Abrecht, 2015) and co-occurrence statistics (Lin and Och, 2004). While ROUGE correlates well with human judgments for extractive summarization, it does not perform as well for abstractive summarization since the chosen new words may not overlap with the reference summary, although possibly preserving the general meaning of the text. Some work has been done in trying to take advantage of dense representations to measure similarity rather than semantic overlap (Ng and Abrecht, 2015), which also generalizes the ROUGE framework for abstractive summarization settings.

If ROUGE essentially measures the recall of the generated sentences (how much of the candidate sentence is in the reference summary), Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) measures the accuracy (how much of the reference summary is in the candidate sentence). Although it originated in machine translation (ranking possible translations), BLEU was applied to summarization in the very first DUC challenge, but subsequent challenges evaluated performance using ROUGE.

7. Related Work

At its core, extractive summarization is a text ranking problem, where we have to choose the most important words to preserve in a final, shorter text. This formulation has a simple translation to SDS: rank and select the parts of the original text that should appear in the summary. However, this simplicity breaks down when we pass to the

[1] `https://duc.nist.gov/` (Accessed January 20, 2021)
[2] `https://tac.nist.gov/` (Accessed January 20, 2021)

multi-document world: the set of documents to summarize may not talk about the same thing at the same level of detail, so we must identify and eliminate some redundancy. Also, we need to ensure that the produced summary is coherent with respect to the different source texts (Radev et al., 2002). A common way to solve this problem is to collapse the task into SDS: just concatenate all the texts. This, however, also creates new problems. News articles, for example, make an effort to have their first sentence be the most prominent (or "summarizable"), so the concatenated document of news articles would have multiple "first sentences" throughout. This also destroys the text's narrative: the content no longer begins in the introduction and ends in the conclusion, it now instead has various phases where it begins and ends anew.

Multi-document summarization is a well-studied field, featuring many different approaches with various degrees of success, most of which fall into the pitfall of modeling it as an SDS problem. Nonetheless, we highlight below some recent or seminal work done in the area.

Goldstein et al. (2000) generalized the Maximum Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) approach to extractive summarization to the MDS setting. MMR is a method for selecting sentences to include in a summary that a) provide new information, and b) are not similar to the already included sentences. Let $s \in D$ be a sentence in a document D, R be the set of sentences already chosen as relevant and that have been selected to appear in the summary, and Q be some user query. MMR is then defined as

$$\begin{aligned}\mathrm{MMR}(D, Q, R) := \mathrm{argmax}_{s \in D \setminus R} \\ \{\lambda \, \mathrm{sim}_1(s, Q) - (1 - \lambda) \max_{s' \in R} \{\mathrm{sim}_2(s, s')\}\}\end{aligned} \tag{14}$$

where $\lambda \in [0, 1]$ is a parameter controlling if we want the selected sentence to be more relevant towards the query (as measured by some similarity metric sim_1) or more diverse (by metric sim_2) towards the already selected sentences. In Goldstein et al. (2000), the authors applied MMR to MDS by incorporating a series of document-independent statistical heuristics, such as number of documents that contain the query, the document where a selected sentence comes from, the timestamps between document publications, among others. Testing was done using the TIPSTER corpus (Harman and Liberman, 1993), evaluated using both compression ratio and cosine similarity to reference human summaries.

Erkan and Radev (2004) introduced LexRank, a graph-based method for ranking TF-IDF (see section 2.1) represented sentences in a text document. LexRank first constructs an undirected graph with nodes representing sentences and edges representing the cosine similarity between sentences (where edges are only present if this similarity is above some threshold). It then applies the concept of eigenvalue centrality in graph-theory, achieved by multiplying the adjacency matrix by some initial distribution over all the vertices in the graph until convergence. A summary is then built by selecting the top n-th sentences, and evaluated using ROUGE-1 on the DUC 2003 and DUC 2004 corpora.

Zhao et al. (2009) did query-focused graph-based MDS extraction by selecting the top sentences that are closest to the query using LexRank. Afterward, these sentences are added to the user query, after which all the sentences are re-ranked according to this new query, paying attention to redundancy. This is done to reduce the information noise in the documents, and, as such, should generate better summaries. Testing was done in DUC 2005 and DUC 2006 using ROUGE-1, ROUGE-2, ROUGE-S, and ROUGE-SU4, where their method is comparable to the top performing systems in the DUC challenge for those corpora.

Kågebäck et al. (2014) explored the viability for summarization of modeling sentences with semantically-aware representations (such as SGNS vectors, see section 2.2). To achieve this, they represented sentences using a simple (sentence vectors are given by the sum of the word vectors) and a complex method (sentence vectors are given by a recursive auto-encoder (Socher et al., 2011) that explicitly models the word order in the sentence and the grammar used). The dataset used is Opinosis (Ganesan et al., 2010) (short user reviews on different topics), evaluated using ROUGE-1, ROUGE-2, and ROUGE-SU4; they concluded that simpler methods (i.e., sentences are the sum of their words) outperform more complex ones.

Yogatama et al. (2015) represented each sentence by a vector given by Latent Semantic Indexing (see section 2.1). Given that a set of vectors (points) forms a polytope, a summary is built by selecting the sentences corresponding to the set of points that form the widest polytope over all the other sentences in the document cluster, the assumption being that the generated summary will be the one with the highest coverage (maximum volume) and least redundancy (points chosen are, by construction, at the boundary – see Figure 2). Their method was evaluated on TAC 2008 and TAC 2009 using ROUGE-1, ROUGE-2, and ROUGE-SU4, and compared against MMR and Coverage-Based Summarization (Gillick et al., 2008), where the generated summaries that cover more diverse bi-grams scored higher.

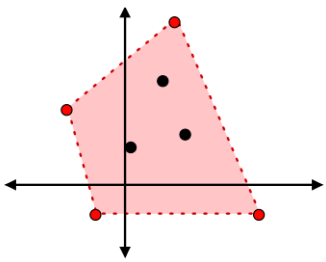

Figure 2: The four red dots represent the selected sentences since the polytope spanned by them covers all the other sentences in the document. Image adapted from Yogatama et al. (2015).

One should note that this is the same as finding the

set of points that span the polytope with the largest area. This is similar to an approach that we explore to construct summaries, with the difference being that we find the set of points that span the smallest volume polytope.

Mani et al. (2018) represented a document by a paragraph vector (Le and Mikolov, 2014) and, for a document cluster, its centroid is seen as the "content average" of all the documents in the cluster. Summarization is done by selecting sentences that minimize the euclidean distance to the centroid of the cluster. They evaluate on DUC 2006 and DUC 2007 using ROUGE-1, ROUGE-2, and ROUGE-SU4. The idea of averaging representations of different objects into a final statistic is close to one of our proposed approaches, the main drawback in the case of Mani et al. (2018) is that their representation model needs to be pre-trained on some other dataset (they chose Thomson Reuters Text Research Collection (Lewis et al., 2004) and CNN/Dailymail (Hermann et al., 2015) corpora).

Lebanoff et al. (2019a) proposed testing the effectiveness of sentence fusion in abstractive summarization settings. To this end they found that current state-of-the-art systems are not doing (either implicitly or explicitly) sentence fusion, ending with the note that sentence fusion is not as effective as previously thought. Evaluation was done using the CNN/Dailymail corpus by human evaluators, where the assessed metrics were a) faithfulness: if the summary remains true to the original text, b) grammaticality: if the summary is grammatically acceptable, and c) coverage: if the summary has information pertaining to selected article highlights. They concluded that the systems that perform fusion rank the lowest on faithfulness and that higher ROUGE scores do not necessarily lead to more faithful summaries. Within the systems that perform fusion, they found that the systems that fuse sentences by simple concatenation are the ones that have the highest faithfulness. In the other metrics, fusion systems are found to generate readable, grammatically correct summaries, but not as much as the state-of-the-art encoder-decoder systems.

Lebanoff et al. (2018) trained an encoder-decoder model to learn how to fuse disparate sentences to generate the summary in an abstractive manner, with an attention mechanism to regulate which sentences to fuse. MMR is also used to calibrate the selection, to account for redundancy in the summary. They evaluate their performance using ROUGE-1, ROUGE-2, ROUGE-SU4 and human judgments on documents from DUC 2004 and TAC 2011, comparing it with the baselines used by Liao et al. (2018) plus an integer linear-programming model for summarization (Gillick et al., 2009). The implications of the mixture of extractive and abstractive summarization is seen as a point to develop further, since, despite not performing as well as some extractive baselines, the summaries generated were more highly ranked by human annotators in faithfulness and coverage. One advanced possibility is that they are not optimizing the extractive part of the method separately from the abstractive one.

Lebanoff et al. (2019b) used vector representations for words that change depending on the context that the word is inserted in (see section 2.2). With this representation, they created a system that compresses or joins two sentences – hence it is an abstractive procedure – selected by an attention mechanism that is sensible to the fact that the sentences may have come from different documents. Tests were done both in SDS and MDS settings, and, for the used MDS corpus (DUC 2004), the baseline extractive methods (see section 8.5, plus N-LEAD, where the first N sentences are taken from each document to form the summary — in their case, N is the average number of sentences in the reference summaries) fared better in ROUGE-1, ROUGE-2, and ROUGE-L scores, indicating that just having the awareness that sentences come from different documents is not enough to properly select important content. Also, for the MDS setting, they find that simpler representations that leverage word frequency across documents (TF-IDF, see section 2.1) outperform more complex ones.

8. Experimental Setup

This section details the datasets, metrics, and steps taken for the summarization algorithm.

8.1. Datasets

We chose to test simplicial curves in the MDS problem using the DUC 2006 and DUC 2007 datasets (c.f. section 5). These datasets were chosen since they are the ones most suitable for MDS, as well as being the ones where most MDS articles have focused on, facilitating comparison of results. Our objective is, given a document cluster, to produce a summary (where summary size varies with the chosen dataset) that is acceptable (measured by some metric — see below) when compared to some human-made reference summary for that document cluster.

8.2. Curve Construction

Every DUC document is first converted from XML to plain text. We chose to construct the curves at three levels: a) the sentence level (each curve represents a sentence); b) the document level (each curve represents a document); and c) the mixed level (each curve represents a document, built by concatenating smaller curves that represent the sentences of that document). Sentences are extracted from every document using Apache OpenNLP[1]. No stopwords were removed and no stemming was done, in order to preserve function words in the generated summaries.

The base matrices were created in two ways: a) 100-dimension vectors from word2Vec; and b) 10-dimension vectors, obtained by applying UMAP (McInnes et al., 2018) to the 100-dimension vectors. UMAP is a dimensionality reduction technique that maintains positional relativity: objects close in the high-dimensional space are mapped

[1] `https://opennlp.apache.org/` (Accessed January 20, 2021)

to close objects in the low-dimensional space and objects further apart in the high-dimensional space are mapped to distant objects in the low-dimensional space..

Curves were built with a smoothing value for the restricted Gaussian kernel of $\sigma \in \{0.05; 0.005; 0.003; 0.001\}$. These values were chosen in-line with the original article. Finding a clear relation between smoothing value and curve performance can be a future area of enquiry. UMAP was run with the default parameters from the authors' implementation.

8.3. Summary Construction

The most straightforward approach to building summaries using curves is the average curve approach. The resulting curve should intuitively model both documents: to get a summary, it suffices to synthesize words from it.

The success of this method is highly dependent on the underlying representation used to build M_y. To summarize a single document, we can also consider curves built with different scales for the kernel, i.e., synthesizing words from the curve $\gamma'_y = \frac{1}{2}(\gamma_y^{\sigma_1} + \gamma_y^{\sigma_2})$ for document y.

We construct the summary curve by a) averaging and b) conflating the curves for all the documents. Reconstructing the text is done by sampling uniform-spaced points from the summary curve and retrieving the word associated with the dimension with the highest probability (in the case of one-hot built curves), or by training an encoder-decoder model to map between curves to sentences.

An LSTM (Hochreiter and Schmidhuber, 1997) was used to create a mapping from curves to sentences in the cases where the curve dimensions do not have any extrinsic meaning. This was done by training a neural-network to match curve representations (dense matrices) of sentences to a vector representation of those sentences. This vector representation has length equal to the length of the sentence, and the entries of the vector are the index positions in the vocabulary of the word in the sentence.

As an example, consider the sentence "How are you, you villain?". If we create a curve from a base matrix representation with 10-dimension vectors, and we sample 5 points of that curve to create a summary, the network will have to map:

$$\mathbb{R}^{5\times 10} \ni \begin{bmatrix} 0.1 & \cdots & 0.15 \\ \vdots & \ddots & \vdots \\ 0.45 & \cdots & 0.03 \end{bmatrix} \longmapsto \begin{bmatrix} 0 & 1 & 2 & 2 & 3 \end{bmatrix}$$

Since any unlabelled collection of texts can be used for this purpose, we trained the model in the DUC 2006 dataset, using only the original documents as the source for our training sentences. The model was constructed using the Keras Framework (Chollet et al., 2015) with the Tensorflow backend. It was trained for 50 iterations using the Sparse Categorical Cross-entropy loss and the RMSprop optimizer. The hidden layer of the LSTM had dimension 100 and its activation function was softmax.

8.4. Evaluation

We compare the generated summaries with the reference summaries using ROUGE-1, ROUGE-2, and ROUGE-L, since these are the most widely used automatic metrics in the MDS task. Although it was also presented in section 6, we do not evaluate our summaries using SERA because this metric has relevance chiefly in the field of summarizing scientific articles, and not general news articles.

8.5. Baselines

We compare simplicial curves with some strong extractive (*ext*) and abstractive (*abs*) baselines that have been applied successfully in multi-document summarization:

- SumBasic (*ext*) (Vanderwende et al., 2007) is a greedy algorithm for sentence selection that chooses to include in the summary the sentence with the highest probability, as given by $P(S) = \frac{1}{\#S}\sum_{w\in S} P(w)$, where $P(w)$ is a unigram distribution of all the words in the corpus. This process is repeated until the desired length of the summary is reached.

- KLSum (*ext*) (Haghighi and Vanderwende, 2009) builds upon the above idea but, instead of including in the summary the sentence with the highest probability, it selects the sentence that, when added to the summary, most reduces the KL-divergence between the unigram distribution of the sentence and the unigram distribution of the corpus.

- TextRank (*ext*) (Mihalcea and Tarau, 2004) is very similar to LexRank (viz. section 7) but, instead of finding the eigenvalue centrality of the sentence graph, sentences are ranked for extraction by the PageRank algorithm. Further, the original TextRank algorithm builds the weighted sentence graph by considering the weight of an edge to be the amount of lexical overlap between two sentences.

- Pointer-Generator (PG) networks (*abs*) (See et al., 2017) is a mixture of extractive and abstractive approaches: when constructing the summary, the model decides (based on an attention mechanism) if, for the current position of the under-construction summary, it is better to generate a new word or to copy a word from the source text. Both the attention mechanism and the underlying model need to be trained using a different corpus from the one we want to summarize.

- PG-MMR (*abs*) (Lebanoff et al., 2018) builds upon the above idea, only that the summary construction step is interleaved with MMR (section 7), where it is used to pick K sentences to pass on to PG. After each summary construction round, the sentences are re-ranked and the process repeats until the summary has the desired length.

9. Results and Discussion

The ROUGE-1, ROUGE-2 and ROUGE-L scores for each baseline and the proposed method are presented in Tables 1 and 2, for the DUC 2006 and DUC 2007 datasets. In the simplicial curves entry is the dimension of the word-embeddings used (10 or 100). "Cat" means that the curve was done at the document level by concatenating curves at the sentence level. The results presented are for curves built with smoothing kernel $\sigma = 0.003$.

DUC 2006	R-1	R-2	R-L
SumBasic	0.27	0.03	0.12
KLSum	0.27	0.03	0.13
TextRank	**0.33**	**0.06**	**0.16**
PG	0.24	0.04	0.13
PG-MMR	0.30	**0.06**	0.15
Simplicial Curves 10	0.02	0.001	0.02
Simplicial Curves 100	0.04	0.001	0.04
Simplicial Curves 100 Cat	0.05	0.004	0.04

Table 1: Baseline and curve results on the DUC 2006 dataset.

DUC 2007	R-1	R-2	R-L
SumBasic	0.29	0.04	0.14
KLSum	0.28	0.04	0.13
TextRank	**0.36**	**0.07**	**0.17**
PG	0.26	0.05	0.14
PG-MMR	0.32	**0.07**	**0.17**
Simplicial Curves 10	0.02	0.001	0.01
Simplicial Curves 100	0.03	0.001	0.03
Simplicial Curves 100 Cat	0.04	0.003	0.03

Table 2: Baseline and curve results on the DUC 2007 dataset.

All presented curve results were obtained by combining curves by average. The conflation method, as discussed in section 4, is not shown as it was not successful: the resulting curve would always concentrate all of its mass around one point, making the curve generate only one word.

We can see that the ROUGE scores achieved for the simplicial curves are not competitive with the baselines. One possible explanation for this is that the curve-averaging method generates curves that are poor information-wise, due to the original curves' distance apart in the 100-dimension word-embedding space (as illustrated in Figure 3, with $\gamma' = \frac{1}{2}(\gamma_1 + \gamma_2)$). This, combined with the fact that the curves pass through the same region many times (because of the stopwords), leads to the resulting average curve being condensed in some specific areas and every so often shooting off into regions with content.

We keep the stopwords because we need the function words to generate legible text. Even if we remove the stopwords, the overall result does not change: the resulting curve now focuses on superfluous words between the curves.

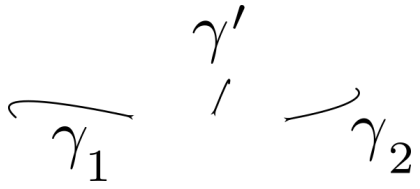

Figure 3: Shortcoming of the curve-averaging method. γ' has no clear relation with γ_1 or γ_2.

One thing to note is the fact that the average curve may pass through regions that do not represent the original documents in any way. To use a standard word-embedding example, if there are two documents that say "hot" and "cold", respectively, then the average curve will pass through the "warm" region, and this would be the word that would be generated for the summary, even though the original documents have no connection to this term. This is also illustrated by Figure 3.

Even so, the ROUGE scores are better if the curves are built with higher dimension base matrices. This is despite the fact that, in higher dimensions, the curves have a much wider space to roam, thus aggravating the above-mentioned shortcoming of the averaging method. We can explain this dissonance in two ways: a) the 10-dimension word vectors were obtained by reducing the dimension of the 100-dimension word vectors with UMAP. This mapping may have rendered the output vectors unfit for purpose, even though UMAP retains object proximity relativity (close objects in higher dimensions remain close in lower dimensions; far away objects in higher dimensions remain far away in lower dimensions); and b) higher dimensions in the word-embedding space can model a higher number of concepts, so the generated words will be richer (this is compatible with the above shortcoming: the curves to average have more dimensions, so the average curve has more regions where it can pass through and not be informative).

It is also interesting to note that constructing document-level curves by concatenating sentence-level curves works better than constructing document-level curves or sentence-level curves by themselves. This is due to the resulting curves being bigger (more information dense) because each part of the curve explicitly models a sentence of the originating document.

Given that the ROUGE results were so poor, we did not use the algebraic machinery developed in section 4. Our original plan was to relate an intrinsic curve feature (e.g., curve entropy) with the quality of the generated summary, as given by the ROUGE scores. However, because these were so low, any correlation score was highly probable to be just noise.

10. Conclusion and Future Work

In this work we have expanded on the concept of simplicial curves (Lebanon et al., 2007), generalizing it to different base representations. We introduced the simplicial curves method and developed an algebra for it, which we did not use in its entirety.

We then explored the effectiveness of the method in the task of multi-document summarization, testing whether or not a representation that explicitly maintains sequencing information is useful in this task. Our experiments show that it is not, with the ROUGE-1 score of 0.04 obtained in the DUC 2007 dataset being below our simplest baseline (SumBasic) score of 0.29, where summaries are constructed simply by selecting the most important sentences in a document. It thus seems that the additional structure provided by the simplicial curves is not being used effectively in generating summaries – essentially resulting in noise.

Some of the tools we had prepared to deal with curves were left unused because of the poor results we had in the MDS task. If the results had been better – at least as good as the most basic baseline – any impact that any upstream change could have had, as guided by the intrinsic evaluation methods we developed, could at least be reliably measured.

Notwithstanding the poor results in the summarization task, we still believe in the potential use for a representation that explicitly encodes sequential information (like, e.g., ELMo (Peters et al., 2018)) and that can be manipulated using standard and advanced tools of mathematics (unlike ELMo). In particular, we would like to explore ways of using curves as an intermediate representation for some domain-specialized downstream algorithms, i.e., either sample the curve for points to use as input or use the curve itself as input.

We would also like to keep exploring the effectiveness of curves in different language-related tasks such as word segmentation or topic modeling, as well as some non-language-related tasks that would nonetheless benefit from having a method for representing objects while preserving sequential information (e.g., video processing).

References

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't Count, Predict! A Systematic Comparison of Context-counting vs. Context-predicting Semantic Vectors. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 1:238–247.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:932–938.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Carbonell, J. G. and Goldstein, J. (1998). The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *SIGIR*, volume 98, pages 335–336.

Chollet, F. et al. (2015). Keras. `https://keras.io`. Accessed November 20, 2020.

Collobert, R. and Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41:391–407.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Ethayarajh, K., Duvenaud, D., and Hirst, G. (2019). Towards Understanding Linear Word Analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy. Association for Computational Linguistics.

Fabbri, A., Li, I., She, T., Li, S., and Radev, D. (2019). Multi-News: A Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Ganesan, K., Zhai, C., and Han, J. (2010). Opinosis: A Graph-based Approach to Abstractive Summarization of Highly Redundant Opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 340–348.

Gillick, D., Favre, B., Hakkani-Tür, D., Bohnet, B., Liu, Y., and Xie, S. (2009). The ICSI/UTD Summarization System at TAC 2009. In *Theory and Application of Categories*.

Gillick, D., Favre, B., and Hakkani-Tür, D. Z. (2008). The ICSI Summarization System at TAC 2008. In *Theory and Applications of Categories*.

Goldstein, J., Mittal, V., Carbonell, J., and Kantrowitz, M. (2000). Multi-document Summarization by Sentence Extraction. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pages 40–48. Association for Computational Linguistics.

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

Harman, D. and Liberman, M. (1993). TIPSTER. `https://catalog.ldc.upenn.edu/LDC93T3A`. Accessed December 28, 2019.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching Machines to Read and Comprehend. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Hill, T. P. (2011). Conflations of Probability Distributions. *Transactions of the American Mathematical Society*, 363(6):3351–3372.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Jones, K. S. (1972). A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, 28:11–21.

Kågebäck, M., Mogren, O., Tahmasebi, N., and Dubhashi, D. (2014). Extractive Summarization Using Continuous Vector Space Models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine*

Learning, pages 1188–1196.

Lebanoff, L., Muchovej, J., Dernoncourt, F., Kim, D. S., Kim, S., Chang, W., and Liu, F. (2019a). Analyzing Sentence Fusion in Abstractive Summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 104–110.

Lebanoff, L., Song, K., Dernoncourt, F., Kim, D. S., Kim, S., Chang, W., and Liu, F. (2019b). Scoring Sentence Singletons and Pairs for Abstractive Summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2175–2189, Florence, Italy. Association for Computational Linguistics.

Lebanoff, L., Song, K., and Liu, F. (2018). Adapting the Neural Encoder-Decoder Framework from Single to Multi-Document Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4131–4141.

Lebanon, G., Mao, Y., and Dillon, J. V. (2007). The Locally Weighted Bag of Words Framework for Document Representation. *Journal of Machine Learning Research*, 8:2405–2441.

Levy, O. and Goldberg, Y. (2014). Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems*, 3:2177–2185.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics (2015)*, 3(1):211–225.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5(Apr):361–397.

Liao, K., Lebanoff, L., and Liu, F. (2018). Abstract Meaning Representation for Multi-Document Summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190.

Lin, C.-Y. (2004). Rouge: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81.

Lin, C.-Y. and Och, F. J. (2004). Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-bigram Statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 605. Association for Computational Linguistics.

Luhn, H. P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*, 1(4):309–317.

Mani, K., Verma, I., Meisheri, H., and Dey, L. (2018). Multi-document Summarization Using Distributed Bag-of-words Model. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 672–675. IEEE.

McInnes, L., Healy, J., Saul, N., and Grossberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3:861.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Mnih, A. and Hinton, G. (2007). Three New Graphical Models for Statistical Language Modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648. ACM.

Ng, J.-P. and Abrecht, V. (2015). Better Summarization Evaluation with Word Embeddings for ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1930, Lisbon, Portugal. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. *EMNLP*, 14:1532–1543.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Radev, D. R., Hovy, E., and McKeown, K. (2002). Introduction to the Special issue on Summarization. *Computational Linguistics*, 28(4):399–408.

See, A., Liu, P. J., and Manning, C. D. (2017). Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Socher, R., Huang, E. H., Pennin, J., Manning, C. D., and Ng, A. Y. (2011). Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word Representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394. Association for Computational Linguistics.

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

Vorhees, E. and Graff, D. (2008). AQUAINT-2. `https://catalog.ldc.upenn.edu/LDC2008T25`. Accessed December 28, 2019.

Yogatama, D., Liu, F., and Smith, N. A. (2015). Extractive Summarization by Maximizing Semantic Volume. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1961–1966, Lisbon, Portugal. Association for Computational Linguistics.

Zhao, L., Wu, L., and Huang, X. (2009). Using Query Expansion in Graph-based Approach for Query-focused Multi-document Summarization. *Information Processing & Management*, 45(1):35 – 41.