Soft Comput (2019) 23:1589–1601
https://doi.org/10.1007/s00500-017-2882-2

METHODOLOGIES AND APPLICATION

Gist: general integrated summarization of text and reviews

Justin Lovinger[1] · **Iren Valova**[1] · **Chad Clough**[1]

Published online: 10 October 2017

Abstract E-commerce is rapidly growing, with review Web sites hosting hundreds of reviews on average for any product. Reading so many reviews is tedious, time-consuming, and with the proposed Gist, unnecessary. We introduce Gist, a system to automatically summarize large amounts of text into informative and actionable key sentences. With unsupervised learning and sentiment analysis, Gist selects the sentences that best characterize a set of reviews. All of this is done in seconds, without prior adjustment or training. Gist extends the current state of the art with a modular system that can take advantage of a priori knowledge and adapt to new domains through easy modification and extension. Gist is a general framework, able to summarize any set of text and easily adapt to specific domains. A robust comparison with state-of-the-art summarization algorithms, on datasets containing hundreds of documents, proves Gist's ability to effectively summarize text and reviews.

1 Introduction

People want to know the value of a product before purchase. Whether a meal at a restaurant, a new movie, or a blender. In the Internet age, online reviews are essential for determining this value. The growth of e-commerce presents another problem: quantity of reviews. Reading reviews requires time and effort from the consumer. With hundreds of reviews, this effort becomes a burden. While aggregate ratings alleviate this problem, they do not provide detailed information. One individual may be concerned with the atmosphere of a restaurant, while another cares only about the quality of the fish they serve. Most Web sites provide only a single overall numeric rating, which gives no information about specific quality factors.

With Gist, we summarize text reviews into a few key sentences that capture the overall sentiment about the product. By presenting natural sentences, we provide easy-to-digest information that is not expressed by overall ratings. To avoid the difficult problem of generating natural language for our text summary, known as text abstraction (Carenini et al. 2013; Fiszman et al. 2004; Hahn and Mani 2000), we use sentences from the text as our summary. This technique is known as text extraction (Carenini et al. 2013; Hahn and Mani 2000; Nenkova and McKeown 2012; Ku et al. 2006; Goldstein et al. 2000) and is currently the premier method in the state of the art (Nenkova and McKeown 2012). Unsupervised learning allows Gist to summarize any product without prior knowledge or training. The result is a generalizing system that effectively summarizes any set of reviews in any language.

Gist extends the state of the art with easy customization and rapid adaptation. Gist is a modular text summarization framework. Rigid systems inevitably lack requirements for specific domains. Rather than attempting to create a catch-all system, we design Gist to allow changes within the existing system while focusing on core features that are essential for text summarization. To that end, we do not implement features that focus on specific parts of speech, but provide a framework, which makes such modification easy.

Section 2 discusses text summarization background and related works. Section 3 provides an overview of the Gist algorithm. Section 4 describes sentiment analysis and how it relates to Gist. Section 5 describes the term frequency,

Communicated by V. Loia.

✉ Iren Valova
ivalova@umassd.edu

[1] Computer and Information Science Department, University of Massachusetts Dartmouth, Dartmouth, Massachusetts 02747, USA

inverse document frequency algorithm for relevance analysis, and how it is used in Gist. Section 6 provides details on extending Gist and presents two minor attributes that we easily add to Gist. Section 7 provides a fitness function utilizing the discussed attributes. Section 8 is a discussion of various metaheuristic search methods, leading to our use of metaheuristic search for Gist. Section 9 examines performance optimizations for Gist. Section 10 compares Gist to state-of-the-art summarization algorithms on datasets containing hundreds of documents, using Rouge. We conclude with Sect. 11.

2 Related works

Hu and Liu (2004) developed a review summarization system that focuses on rating features with association rules and sentiment analysis (Yi et al. 2003; Wilson et al. 2005; Pang and Lee 2008, 2004). Microsoft developed a similar system for restaurant reviews presented in Nguyen et al. (2007). Li et al. (2010) used syntactic tree structures and conditional random fields to achieve similar review summarization. These systems take a structural approach to review summarization, explicitly analyzing text to discover features before summarizing these features. The result is similar to a spreadsheet of feature/value pairs. When features are properly extracted, this system is effective. However, this makes feature extraction a weak point that results in odd or confusing summarization when a set of reviews is not structured as expected. This rigid system is also difficult to adapt to specific considerations and domains. With the proposed Gist, we take a more flexible approach that avoids the difficult and error-prone task of feature extraction.

Although we focus on review summarization, the Gist framework supports general text summarization and allows easy modification and extension for any text domain. While review summarization systems are rare in the state of the art, text summarization is plentiful. Supervised learning methods for text summarization gained popularity in the early 2000s with the work of Turney (2000). While the supervised approach still finds use, unsupervised methods quickly became the state of the art when the TextRank algorithm (Mihalcea and Tarau 2004) proved versatile and effective. Recent works focus primarily on improving performance in specific domains (Rastkar et al. 2014; Chua and Asur 2013) or multi-document summarization (Carenini et al. 2013), rather than exploring new approaches to summarization.

State-of-the-art summarization is dominated by two practical tools: MEAD (Radev et al. 2004; Saggion and Poibeau 2013) and SUMMA (Saggion and Poibeau 2013; Saggion 2008, 2014). MEAD is a text summarization framework, within which numerous particular algorithms are implemented, such as MEAD* (Carenini et al. 2013; Gerani et al. 2014) for multi-document summarization and MEAD-LexRank (Gerani et al. 2014) implementing the popular LexRank algorithm (Erkan and Radev 2004). SUMMA is a set of text summarization resources built on the GATE (Maynard et al. 2002) text summarization framework.

The Gist framework plays to the requirements of modern summarization by allowing easy and rapid adaptation to any domain. We also note that many of these summarization techniques can be easily incorporated into the Gist framework and work alongside or instead of the core attributes presented in the following sections.

3 The Gist algorithm

Gist is composed of a few disjoint components (presented in Fig. 1) that effectively summarize when brought together. The core of Gist is a set of attributes calculated from and assigned to each sentence from the set of text or reviews being summarized. Each attribute is a numeric value calculated from the text in a sentence. A sentence is defined as a sequence of characters followed by a stop character like

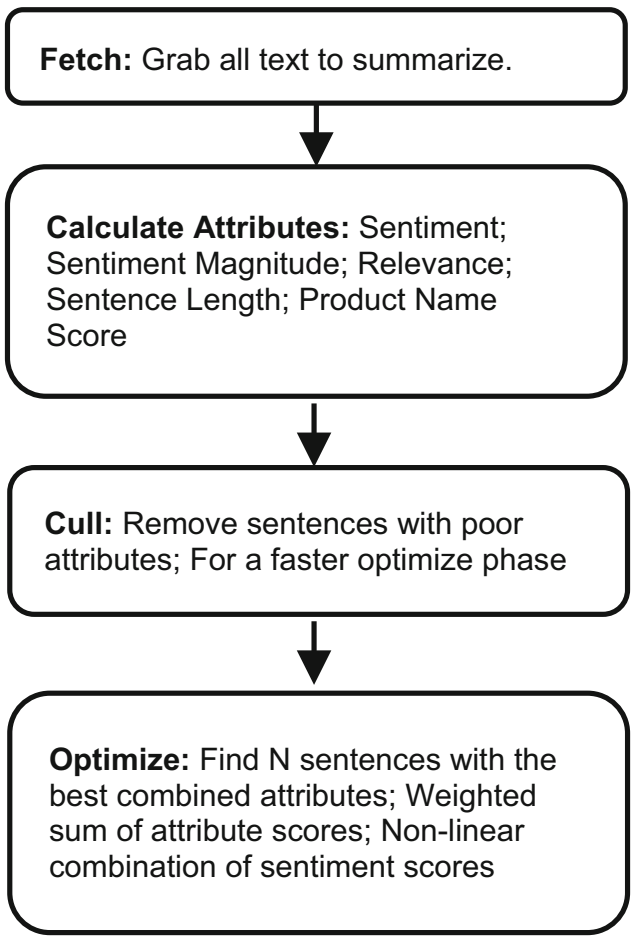

Fig. 1 Phases of Gist

Table 1 Calculated attributes of example sentences

Sentence	Sentiment (s)	Sentiment magnitude (m)	Relevance (r)
s_1	0.4	0.4	0.6
s_2	−0.4	0.4	0.5
s_3	0.7	0.7	0.9
s_4	−0.1	0.1	0.6
s_5	−1.0	1.0	0.3

Table 2 Comparison of attributes and culling thresholds

Sentence	s	m	$m \geq ?t_m$	r	$r \geq ?t_r$	Culled?
s_1	0.4	0.4	$0.4 \geq 0.2$	0.6	$0.6 \geq 0.5$	
s_2	−0.4	0.4	$0.4 \geq 0.2$	0.5	$0.5 \geq 0.5$	
s_3	0.7	0.7	$0.7 \geq 0.2$	0.9	$0.9 \geq 0.5$	
s_4	−0.1	0.1	$\mathbf{0.1 \ngeq 0.2}$	0.6	$0.6 \geq 0.5$	Culled
s_5	−1.0	1.0	$1.0 \geq 0.2$	0.3	$\mathbf{0.3 \ngeq 0.5}$	Culled

Table 3 Calculated fitness of sentence combinations

Sentences	Fitness equation	Fitness
(s_1, s_2)	$10 \times 0.96 + 3 \times 0.40 + 6 \times 0.55 =$	14.12
(s_1, s_3)	$10 \times 0.64 + 3 \times 0.55 + 6 \times 0.75 =$	12.55
(s_2, s_3)	$10 \times 0.89 + 3 \times 0.55 + 6 \times 0.70 =$	**14.76**

period, exclamation mark, or question mark, then followed by whitespace. Of these attributes, the two most important are sentiment and relevance. However, any number of attributes can be assigned to sentences, and special attributes can be calculated for specific domains to utilize a priori knowledge. With these sentence attributes, we perform metaheuristic search with a fitness function that examines the attributes of N sentences and selects the N sentences that best summarize the set of text. These N sentences are then returned as the summary. The ability to add attributes with minimal effort allows one to easily extend Gist for new domains or specific considerations. For example, if trying to summarize the atmosphere of a restaurant, one can insert an attribute that gives extra fitness to sentences that mention atmosphere. For clarity, we provide a step-by-step example:

1. *Fetch* This step simply obtains the text and is domain specific. In this example, the text consists of only 5 sentences. For demonstration purposes, these imaginary sentences are simply referred to by the labels $s_1 \cdots s_5$. Real text can easily contain hundreds of thousands of sentences.
2. *Calculate attributes* Table 1 displays the calculated attributes for our example sentences. For this demonstration, attribute values are fabricated. The following sections explain how these attributes are calculated for real sentences.
3. *Cull* Table 2 compares each sentence's attributes to the given threshold values. The sentiment magnitude threshold $t_m = 0.2$, and the relevance threshold $t_r = 0.5$. Note that sentiment has no culling threshold. All attributes of a sentence must be greater than or equal to the corresponding thresholds, or the sentence is culled. We see that sentence s_4 is removed due to low sentiment magnitude m and s_5 is removed due to low relevance r.
4. *Optimize* For this example, our goal is to find 2 sentences from the text that most closely match the overall sentiment of the text, which we define as -0.2 for this demonstration while maximizing relevance and sentiment magnitude. This goal is formalized as the fitness function $f(N) = 10C(N) + 3m(N) + 6r(N)$, where N is a set of sentences, $C(N)$ is the sentiment closeness score (1) defined in the next section, $m(N)$ is the average sentiment magnitude of N, and $r(N)$ is the average relevance of N. Note that this goal is easily modifiable. Table 3 displays the fitness value and calculation for each sentence not culled. Although the set of sentences (s_1, s_2) most closely fit the overall sentiment, the greater relevance of s_3 results in a higher fitness score for (s_2, s_3). A metaheuristic search algorithm will explore the space of sentence combinations, to find the set with the highest fitness. For this example, we can easily see that the set (s_2, s_3) is the best combination.

4 Sentiment analysis

At its core, sentiment analysis (Yi et al. 2003; Wilson et al. 2005; Pang and Lee 2008, 2004) is a dictionary, which maps words to sentiment polarity. Words like "great", "good", and "awesome" have a positive sentiment, while words like "terrible" and "bad" have a negative sentiment. Many words have no sentiment at all, such as "the", "and", "a". The greatest difficulty lies in generating this dictionary. In this paper, we do not focus on the generation of a sentiment dictionary. For Gist, we use the expansive WordNet lexical database (Miller 1995). Beyond the core sentiment dictionary, pattern analysis provided by the TextBlob library (Loria 2014) allows for more accurate sentiment classification by recognizing linguistic constructs such as negation, and in some cases sarcasm (González-Ibánez et al. 2011).

It is worth discussing the disadvantages of sentiment analysis, both in the current state of the art and in general. First, we make no claims to the optimality of our sentiment analysis. Certain forms of speech, such as sarcasm and unusual use of words, can result in a misclassification of sentiment. Even if our sentiment analysis is perfect, the issue of subjectivity arises. To perfectly classify sentiment, the analysis must tailor its behavior to an individual. Language is a com-

plex construct, and everyone interprets words and sentences differently. A sentence that is very positive to one individual could be negative to another, based on culture or personal experience. Even human judges perform poorly at sarcasm classification (González-Ibánez et al. 2011). As it is unreasonable to tailor our sentiment analysis to every individual, we must settle for an imperfect system. However, since sentiment analysis in Gist acts only as heuristic guidance, Gist can function well in the intended environment.

4.1 Sentiment attribute in Gist

Sentiment analysis plays a key role in Gist. Reviews are about feeling. One review may have great things to say about a product, which is a positive sentiment. Another may hate the product, which is a negative sentiment. Our first step to summarizing a set of reviews is determining the overall sentiment of the reviews. Note that the sentiment analysis procedure described in Sect. 4 can be applied to the entire set of text to obtain the overall sentiment. Next, we obtain the sentiment for every sentence. By examining the average sentiment of a set of N sentences, we can determine how close the N sentences match the overall sentiment of the product. As such, a product with generally positive reviews will have a positive summary, and vice versa. This is formalized in Eq. (1):

$$C(N) = \frac{1}{\left(P - S_{\text{avg}}(N)\right)^2 + 1}, \tag{1}$$

where P is the overall product sentiment and $S_{\text{avg}}(N)$ is the average sentiment of a set of N sentences. With this function, $C = 1$ when average sentiment matches P. As average sentiment diverges from P, C approaches 0.

We also examine sentiment magnitude, the absolute value of sentiment. This heuristic leads us to sentences that are well opinionated. When generating a summary, we want sentences that tell us something about the product, which is to say, we want opinions. A sentence such as "I ordered the chicken" tells us less than "The chicken was wonderful".

5 TF–IDF

The term frequency (TF) component of TF–IDF dates back to 1957 with Luhn (1957). The concept is simple, the more a word is used in a set of text, the more relevant it is to that set of text. This is formalized in Eq. (2):

$$\text{TF}(w) = (\text{count of } w \text{ in } T)/(\text{count of terms in } T), \tag{2}$$

where w is a word (or term) and T is a set of text. Term frequency alone would give disproportionate weight to commonly used words such as "and", "the", "a". Even though these words show up frequently in any set of text, they are not relevant because they are simply necessary grammar. The fact that they are omnipresent allows us to solve this problem with inverse document frequency (IDF), first introduced in Sparck Jones (1972). Like TF, IDF is conceptually simple. The more documents a term appears in, the less relevant it is for any particular document. This is formalized in Eq. (3):

$$\text{IDF}(w) = \ln(\text{count of documents}/\text{count of documents containing } w). \tag{3}$$

If a term appears in every document, its IDF value is 0, while a term that appears in only one document can easily have a value in the double digits. Combining these two concepts, we obtain the TF–IDF algorithm:

$$\text{TF} - \text{IDF}(w) = \text{TF}(w) \times \text{IDF}(w). \tag{4}$$

Conceptually simple, and dating back to the mid-1990s, this algorithm is proven effective and timeless, with numerous examples of modern usage (Chum et al. 2008; Zhang et al. 2011; Arandjelović and Zisserman 2012).

5.1 Relevance attribute in Gist

If we only examine sentiment when generating a summary, an issue arises. For the "McDonalds" restaurant, the summary could include "Star Wars is a really great movie". This sentence is completely irrelevant to the restaurant. To eliminate such sentences, we use the TF–IDF algorithm given in Eq. (4). For our document corpus, we use the well-tested Reuters-21578, Distribution 1.0 (Lewis 1997; Debole and Sebastiani 2005), containing 21578 documents.

The core of TF–IDF is a dictionary of word values. To obtain this dictionary, we must calculate the TF value of every word in the set of reviews we are summarizing (Eq. 2) and the IDF values for our corpus of documents (Eq. 3). Next, using Eq. (4), we obtain the relevance R for sentence S with m words, as in Eq. (5):

$$R(S) = \frac{\sum_{w \in S} \text{TF} - \text{IDF}(w)}{m}. \tag{5}$$

In short, $R(S)$ is the average TF–IDF score of all words, w, in sentence S. Normalizing by the number of words, m, allows us to obtain a relevance value independent of the size of the sentence, thereby providing more effective relevance comparison. This attribute allows us to penalize outlier sentences and reward sentences that closely match the average language used in the set of reviews.

6 Extensibility

Review summarization is a narrow view of the possible applications of Gist. The ability to easily add attributes allows Gist to extend to other domains without reinventing the core system. General text summarization is already possible with the core Gist attributes.

For example, one can adapt Gist to extract sentences mentioning recent news from social media. Each set of text can be taken from different social media posts, or from many posts from each person. An attribute that gives great weight to sentences containing phrases related to recent news, and a filter for these phrases extend Gist toward the domain of news summarization. Combined with the core Gist attributes, this system could find sentences that fit people's views of recent news. If interested in only positive or negative views, a corresponding sentiment filter can be added.

We note that, while the formulation in Fig. 1 and Sect. 7 specifies a linear combination of attributes, nonlinear attributes are possible. Attributes can be calculated from a set of sentences, during the optimization phase, at the expense of performance. The sentiment attribute in Sect. 4.1 is an example of a nonlinear Gist attribute.

It is also easy to improve Gist's effectiveness as a general text or review summarization system. As an unsupervised learning system, the effectiveness of Gist summarization is determined by user opinion. Although we have limited resources to thoroughly test the effectiveness of various combinations of attributes, if more are added, user opinion can be obtained on whether the attributes improve summarization ability. This step could be repeated multiple times to optimize text summarization. Giving greater weight to sentences with certain parts of speech is an example of an attribute that may improve text summarization.

6.1 Minor attributes in Gist

Beyond the primary attributes, a number of minor attributes provide additional direction for Gist. For one, Gist grants additional fitness to sentences that mention the name of the product, or to a lesser extent, part of the name. This reduces the chance of generating a summary with sentences that are out of context. The sentence "Star Wars is a really great movie" is given greater weight than "That is a really great movie".

By examining the length of a sentence, we provide heuristic guidance toward sentences that fit the concept of a summary. Short sentences are penalized for not providing enough information, while long sentences are penalized for being too verbose. The optimal sentence length is empirically determined to be 15 words. This conclusion is reached by calculating the average sentence length from hundreds of reviews that are deemed very helpful by the Yelp community.

Table 4 Attribute weights

Attribute	Weight
Sentiment, C(N)	10
Relevance	6
Sentiment magnitude	3
Length score	2
Name score	1

7 Fitness function

Each attribute presented in Sects. 4–6 provides a value correlating with the effectiveness of a sentence or set of sentences for summarization. These attributes consider sentiment, relevance, length, and title (name) of the document or product. Further attributes can be easily added to consider other aspects of a good summary. Because each attribute individually scores the effectiveness of a sentence or set of sentences for summarization, combining these attributes with weighted summation provides an effective fitness function.

This fitness function must first combine individual sentence attributes into a combination value for each attribute. Some attributes are combined as an average of individual sentence attributes; others are a nonlinear combination. A weighted summation of attribute combination values then provides the final fitness value. Our attribute weights are given in Table 4. Note that each attribute is defined in Sects. 4–6. This weighted sum benefits from reliable value ranges. Apart from relevance, all attributes fall within a predictable range. For relevance values to be more predictable, we normalize from 0 to 1. Algorithm 1 presents this fitness function for a set of N sentences.

Algorithm 1 Gist Fitness Function

```
Set fitness F = 0
for attribute A, excluding sentiment do
        F = F + W(A) * A_avg (N)
end for
F = F + W(C) * C(N)
```

W(A) is the weight of attribute A, W(C) is the weight of the sentiment closeness attribute, and A_{avg} (N) is the average value of attribute A for sentences in N. Of our attributes, only the sentiment closeness function C(N) (1) is a nonlinear combination. As described in Sect. 6, Gist can be extended with additional nonlinear attributes.

Since attributes are pre-calculated for every sentence, the fitness function is very fast, requiring only an addition operation for every attribute of every sentence and a multiplication operation for every attribute, with the exception of C(N), which scales linearly with the size of N. Fitness function scaling is formalized as O (ak), where a is the number of attributes and k is the size of N.

8 Metaheuristic search

The ability to traverse a general fitness landscape makes metaheuristics a powerful search tool. Each metaheuristic uses the same fitness function, defined in Sect. 7. This fitness function takes a set of sentences and returns a single number that defines the quality of the given sentences with regard to summarizing the text. In Gist, this fitness value is determined by a combination of sentence attributes such as sentiment, TF–IDF relevance, and other values defined by a user. The generality of metaheuristic search lets us change the definition of a good sentence by adjusting the fitness function, without modifying the process that discovers the best sentences for summarization.

Many metaheuristics exist to perform search. We explore a popular subset of metaheuristics. The classic genetic algorithm (GA) remains an effective search method (Deb et al. 2002; Yang and Honavar 1998; Uğuz 2011). More recently, metaheuristic methods rooted in mathematics have grown in popularity. Particle swarm optimization (PSO) (Kennedy and Eberhart 1995) is now a staple of metaheuristic methods. The recently developed gravitational search algorithm (GSA) (Rashedi et al. 2009) is functionally similar to PSO, but is inspired by physics rather than nature. The Gist fitness function works with most metaheuristics. As such, the choice of metaheuristic affects only the efficiency of the search for the best set of sentences. This is an interchangeable component that does not affect the core process of Gist.

8.1 Genetic algorithm

The GA (Deb et al. 2002; Yang and Honavar 1998; Uğuz 2011), inspired by evolution in nature, is one of the oldest metaheuristics. Despite its age, this algorithm is one of the most popular metaheuristics in use today, a testament to its effectiveness. One advantage of the GA is extensibility. The use of interchangeable crossover, selection, and mutation functions allows one to tailor the GA to a particular problem.

Selecting candidate solutions, or chromosomes, based on fitness creates pressure to improve fitness and forms the core of the exploitation component of this metaheuristic. Selected chromosomes are then combined to explore potential solutions in the portion of the problem space that high-fitness solutions occupy, adjusting potential solutions toward higher fitness and exploring new space. Mutation maintains diversity in the chromosome population by randomly adjusting chromosomes independent of fitness.

In this paper, we do not attempt to improve upon the GA literature. Instead, we make use of the staples already developed. One-point crossover takes two parent chromosomes and generates two children chromosomes by taking one part from each parent, according to a randomly selected crossover point, displayed in Fig. 2.

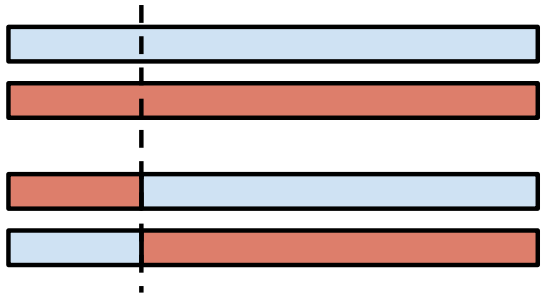

Fig. 2 One-point crossover

Roulette selection randomly selects parents for crossover, with a probability of selection proportional to the fitness of the chromosome. Bit string mutation examines every bit of every child chromosome and, with probability p, flips the bit from 0 to 1, or 1 to 0. Conversely, with probability $(1 - p)$ the bit remains unchanged.

8.2 Particle swarm optimization

Like GAs, PSO (Kennedy and Eberhart 1995) is inspired by nature, but is less dependent on controlled randomness. By studying the swarming behavior of birds and fish, Kennedy and Eberhart (1995) developed this algorithm to mimic the social behavior of animals that can scatter, change direction suddenly, and regroup to discover an optimal path.

PSO initializes a population of particles at random positions in a problem space. Random acceleration vectors drive these particles to explore new solutions. Acceleration toward high-fitness particles, and the best known solution, allows for the exploitation of known solutions in an effort to find small adjustments that increase fitness. Figure 3 shows how random acceleration and acceleration toward high-fitness areas may combine to give the movement of low-fitness particles through a problem space.

Unlike GA, PSOs small adjustments of a previous population allow for fine-grained exploration of continuous problem spaces. Even without a continuous problem space, PSOs behavior causes low-fitness particles to quickly swarm toward high-fitness solutions, allowing rapid adjustment. However, this high exploitation comes at the cost of potentially converging to local minima.

8.3 Gravitational search algorithm

GSA (Rashedi et al. 2009) has many similarities to PSO. Like it, iterative adjustments are made to an initially random population, as opposed to generating completely new solutions from iteration to iteration as GA does. Also like PSO, GSA draws low-fitness solutions toward high-fitness solutions as a form of exploitation. Unlike PSO, GSA is inspired by the

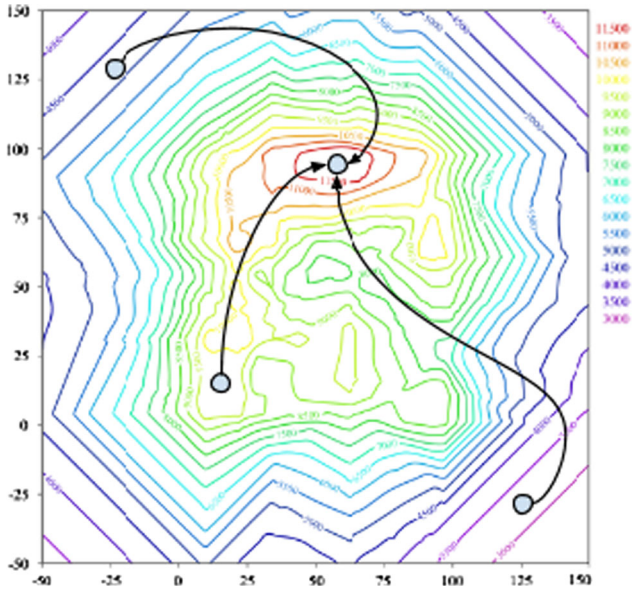

Fig. 3 Particles drawn toward high fitness (https://commons.wikimedia.org/wiki/File:Contour2D.svg#/media/File:Contour2D.svg)

gravitational motion of bodies in free space, as dictated by Newton's laws.

In GSA, every potential solution has a mass proportional to its fitness. As in natural physics, bodies attract one another, higher-mass bodies attract more and are less affected by the attraction of other bodies. This mechanism alone would quickly cause all bodies to converge toward the highest fitness bodies. Unless the global optima lie along the path of one of these bodies, this would result in premature convergence to a local optimum. To encourage exploration of the problem space, the summation of force on a body and the velocity update for a body is randomized. Specifically, the total force on a body in one dimension is given as:

$$F_i^d(t) = \sum_{j=1}^{N} rand_j F_{ij}^d(t), \quad \text{where} j \neq i, \tag{6}$$

where $F_i^d(t)$ is the total force on body i in dimension d at time step t, $rand_j$ is a random number between 0 and 1, $F_{ij}^d(t)$ is the force of body j on body i in dimension d at time step t, and N is the number of bodies. As such, the force that each body applies to every other body in each dimension is randomized. Depicted in Fig. 4 are two bodies of equal mass, equidistant in the x and y dimensions. The force for each dimension is equal according to Newton's laws, unlike GSA.

The velocity update is likewise randomized for each body by taking a random fraction of the velocity in each dimension and adding its acceleration in that dimension, to obtain the

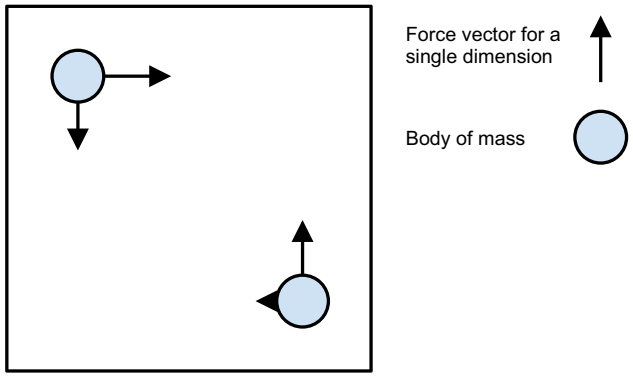

Fig. 4 Movement of bodies in GSA

velocity for the next time step. This is formalized as:

$$v_i^d(t+1) = rand_i v_i^d(t) + a_i^d(t), \tag{7}$$

where $v_i^d(t)$ is the velocity of body i in dimension d at time step t, $rand_i$ is a random number between 0 and 1, and $a_i^d(t)$ is the acceleration of body i in dimension d at time step t. Presented here are the equations that most differentiate GSA from standard Newtonian motion. For a more in-depth presentation of the GSA algorithm, see Rashedi et al. (2009).

9 Performance enhancements

The system presented thus far relies on a metaheuristic without guarantee of finding the optimal solution in a potentially large problem space. This means, for very large problems, if most sentences are poor with regard to the fitness function, the summary may be poor. To more consistently discover the optimal set of sentences, we must reduce the problem space by culling poor sentences, thereby optimizing the metaheuristic.

9.1 Culling

Every sentence adds to the search space of the metaheuristic search process. However, many sentences can easily be deemed unsuitable due to the value of a single attribute. Completely neutral, irrelevant, or extremely long sentences can immediately be discarded from the search space, thereby saving time and increasing the chance of finding the optimal set of sentences.

To this end, we introduce a technique we call intermediate culling. As each attribute A of sentence S, given as A(S), is calculated, it is compared to a threshold. If the attribute value does not pass the threshold, the sentence is immediately discarded. No additional attributes are calculated for the discarded sentence. Otherwise, the next attribute is calculated, and this process repeats. This is presented in Algorithm 2.

Algorithm 2 Calculating Sentence Attributes With Culling

```
fetch all text T
for sentence S in T do
    for attribute A in set of attributes to calculate do
        calculate A(S)
        if A(S) < culling threshold for A do
            stop calculating attributes for S
            remove S from set of sentences
            skip to next sentence in T
        end if
    end for
end for
```

The culling algorithm requires only a single pass of the sentence dataset. In the worst case, when no sentences are culled, performance scaling is given as O (sa), where s is the number of sentences and a is the number of attributes. In actuality, average performance is better, due to attributes not passing the threshold, resulting in fewer than a attributes being calculated for some sentences. Average complexity will depend on the culling thresholds and the nature of the sentence dataset.

9.2 Metaheuristic comparison

In this section, we present our performance results for each of the following metaheuristics: canonical genetic algorithm (GA), particle swarm optimization (PSO), and gravitational search (GSA). Each algorithm runs for 1000 iterations with the best fitness recorded. Since these metaheuristics are stochastic, each test is run 1000 times and the results are averaged. Standard deviation (SD) is also presented to determine consistency of performance. This test is performed on three different sets of restaurant reviews. One-way analysis of variance (ANOVA) (Iversen and Norpoth 1987; Arcuri and Briand 2014) is also presented to ensure statistical significance of these results.

Each algorithm has a population size of 20. The GA has a mutation rate of 0.02, a crossover rate of 0.7, a standard roulette selection function, and uses one-point crossover. PSO has an ω value of 0.5, a ϕ_p value of 0.5, and a ϕ_g value of 0.5. Where ω is a scaling factor for particle velocity, ϕ_p is the pull of a particle's best known position and ϕ_g is the pull of the global best known position. GSA has an initial G value of 1 and a G reduction rate β of 0.5. The GA uses a binary chromosome of size $ceil\left(log_2 S\right) \times N$, where S is the number of sentences after culling and N is the number of sentences to include in the summary. To obtain sentences from this binary chromosome, a block of binary is decoded into an integer index N times. If an index is out of bound

Table 5 Statistical significance of metaheuristic comparison (ANOVA)

Review set	F value	p value < 0.05
Reviews 1	1205.13	$0.0000 < 0.05$
Reviews 2	1664.55	$0.0000 < 0.05$
Reviews 2	1037.11	$0.0000 < 0.05$

Table 6 Metaheuristic comparison

	Reviews 1	Reviews 2	Reviews 3	Mean
GA				
Mean best fitness	17.91	17.82	15.79	17.17
Best fitness SD	0.522	0.393	0.303	0.197
PSO				
Mean best fitness	17.61	17.45	15.46	16.84
Best fitness SD	0.543	0.581	0.370	0.293
GSA				
Mean best fitness	16.72	16.65	14.97	16.11
Best fitness SD	0.506	0.309	0.256	0.214

Table 7 Summarization algorithm comparison

Dataset		Algorithm		
		Gist	TextRank	LexRank
Opinosis	Avg F-score	0.22093	0.193148	0.150139
	Avg recall	0.4212	0.495914	0.570195
	Avg precision	0.153134	0.121997	0.089143
	Avg runtime	0.059655	0.366839	0.616853
DUC 2004	Avg F-score	0.136521	0.120366	0.098491
	Avg recall	0.324198	0.385268	0.494476
	Avg precision	0.09108	0.071899	0.055598
	Avg runtime	0.027787	0.056858	0.038793
cmp-lg	Avg F-score	0.276999	0.292668	0.294835
	Avg recall	0.271538	0.286066	0.46458
	Avg precision	0.31577	0.340986	0.237358
	Avg runtime	0.067235	0.541721	1.383484

for the array of sentences, the chromosome is immediately given a very low fitness. The PSO and GSA use a solution vector of N real numbers, which can easily become indices by taking the floor of each real number.

The p values in Table 5 show high statistical significance in the difference between GA, PSO, and GSA, on each set of reviews, as given by the low p-values. As such, we can confidently state that the variation between performance means in Table 6 is due to the algorithms themselves, and not random variance. We note that, although ANOVA assumes normality of data and equality of variance, the central limit theorem states that, for large sample sizes, parametric tests like ANOVA are robust even if the data violate these assumptions (Arcuri and Briand 2014; Rice 2006; Sawilowsky and Blair 1992).

As shown in Table 6, GA achieves the highest fitness for all sets of reviews, with highest consistency, as indicated by the lowest standard deviation (SD). However, PSO and GSA follow closely. While PSO and GSA excel at exploring continuous problem spaces, the discrete nature of this problem voids that advantage. GA on the other hand is designed for discrete problem spaces such as the Gist problem tested here. GA also has higher exploration, giving it an advantage in this large problem space. PSO and GSA excel in problem spaces with smooth gradients, but selecting sentences results in a problem space resembling a complex step function with many basins of attraction. These basins easily trap PSO and GSA, while the mutation operators of GA allow it to escape and continue the search for the optimal solution. These factors favor the time-tested performance of GA, making it most effective at determining the best sentences to form a summary from a set of text.

This analysis leads us to implementing GA as the metaheuristic search component of Gist. However, Gist is naive to its particular metaheuristic algorithm. As long as a metaheuristic is compatible with the fitness function given in Algorithm 1, it will function with Gist. Given the vast number of metaheuristics developed, we do not perform an exhaustive comparison. Instead, we use it as guidance when selecting a metaheuristic for Gist. In the classic trade-off of exploration vs exploitation, GA has greater rate of exploration than PSO and GSA. This proves advantageous for the large problem space Gist must search. GA also searches a discrete problem space, while PSO and GSA excel at continuous problem

Table 8 Movie review summaries

Movie title	Star Wars: Episode IV – A New Hope	The Matrix	Saving Private Ryan
Gist summary	The plot was very interesting for it's time, and still is today	Very good action movie, with good story!	It will renew your faith in mankind, while simultaneously horrifying you at man's folly
	Star Wars is still, in my opinion, the greatest film in this series and of all time	If you haven't seen The Matrix series, I pity you, for you have missed the BEST movie ever made, go see it NOW!	I would certainly recommend this movie to anyone that can handle some graphic, painful, bloody action
	Without a doubt this is not just the best star wars movie but also the best sci-fi film ever made	The perfect illustration of control and power and how it influences everything around you	The best war movie ever made
TextRank summary	Great action scenes, story and mind blowing special effects for the time(They still look good today).Without a doubt this is not just the best star wars movie but also the best sci-fi film ever made	All in all this film delivers great any day entertainment that is like Laurence Fishburn said, "Unfortunately no one can be told what the Matrix is you have to see it for yourself." And trust me once you do you won't ever look at Sci-Fi or other movies the same way again	Bookended by the most shocking, searing battle sequences in film history, Saving Private Ryan is as powerful, devastating, memorable and moving as movies get
LexRank summary	It's Star Wars…	The Matrix	It's a reminder that, after all, "Saving Private Ryan" is only a movie
	It's Star Wars	A mind-bending movie	When he looks at Ryan (and the camera is just over his should, so he is basically looking right at the camera) and says "Earn this," he is saying that to all of us
	Star Wars	The Matrix!	Saving Private Ryan was such an amazing movie

Table 9 Pony Express summaries

Title	Pony Express Moving Services
Text	These guys are the best. They quoted my move at $1100 and planned ample man power and truck space for the move. They were very professional, fast and efficient. To my surprise, the move only took 3 hours to load and unload (including drive time) and my bill was only $500! They arrived at my apartment in Somerville at 8:30 and I was moved in by 10:30 in JP. Pony Express is the best moving company I've dealt with. Give them a call!
	Pony Express was amazing. They did great work & were very efficient. I felt complete trust in their ability with my belongings. We moved 45 mins away and nothing was broken or missing or damaged by the end of it. Success! Will use them again
	This was my third move with Pony and they did their usual terrific job. Everything was wrapped and packed quickly but carefully. The guys were pleasant and couldn't have been more helpful once we got to the destination. It all took much less time than expected. They're truly pros!
Gist summary	Pony Express is the best moving company I've dealt with
	We moved 45 mins away and nothing was broken or missing or damaged by the end of it
	Pony Express was amazing
TextRank summary	These guys are the best
	They were very professional, fast and efficient
	Pony Express is the best moving company I've dealt with
	Pony Express was amazing
	They did great work & were very efficient
	It all took much less time than expected
LexRank summary	To my surprise, the move only took 3 hours to load and unload (including drive time) and my bill was only $500!
	Pony Express is the best moving company I've dealt with
	This was my third move with Pony and they did their usual terrific job

Table 10 Keepsake summaries

Title	Keepsake
Text	Not a hidden gem, more like hidden silver piece but I'm still glad that I found it. Atmosphere was great however the hint system was a bit too tempting but you gonna need it eventually
	it is a nice game but there is too much talking and technically it is like going back 5 years ago even more and the ending is so full of cliche. Allways the same sound of footsteps or door opening and closing whatever the place. By chance the puzzles were good!
	I give it a 3 because the puzzles weren't bad and the graphics and sound were passable. This game fails in every other way. The story, voice acting, pacing, gameplay, characterization and the presence of bugs are all disappointing. Comparisons to The Longest Journey can't be serious. (I haven't tried Syberia.) I might recommend this game to a 9 year old or a senior citizen who is unfamiliar with computers, but even then I'd have my doubts
Gist summary	Not a hidden gem, more like hidden silver piece but I'm still glad that I found it
	By chance the puzzles were good!
	This game fails in every other way
TextRank summary	Allways the same sound of footsteps or door opening and closing whatever the place
	By chance the puzzles were good!
	I give it a 3 because the puzzles weren't bad and the graphics and sound were passable
	This game fails in every other way
LexRank summary	Not a hidden gem, more like hidden silver piece but I'm still glad that I found it
	I give it a 3 because the puzzles weren't bad and the graphics and sound were passable
	I might recommend this game to a 9 year old or a senior citizen who is unfamiliar with computers, but even then I'd have my doubts

Table 11 Clever Surveys summaries

Title	Clever Surveys
Text	Personalized predictions from survey answers. Using cutting edge ai, our predictors answer your questions
	Need help choosing a college major or deciding what class to play in your favorite game? Predictors are designed to make your decisions easier. Every predictor is an expert on a particular topic. Answer some questions, and it will give you its expert opinion
	Get Predictions
	Have a pressing question or hard decision? At Clever Surveys, you can create your own predictors. Just follow the link below. Once the predictor learns from people answering a survey, it will be ready to answer your question
	Build Your Own Predictor
Gist summary	Using cutting edge ai, our predictors answer your questions
	Once the predictor learns from people answering a survey, it will be ready to answer your question
	At Clever Surveys, you can create your own predictors
TextRank summary	Personalized predictions from survey answers
	Using cutting edge ai, our predictors answer your questions
	At Clever Surveys, you can create your own predictors
	Once the predictor learns from people answering a survey, it will be ready to answer your question
LexRank summary	Personalized predictions from survey answers
	Answer some questions, and it will give you its expert opinion
	Once the predictor learns from people answering a survey, it will be ready to answer your question

spaces. As such, we conclude that high-exploration, discrete metaheuristics function best with Gist.

10 Benchmark and comparison

Rouge (Lin 2004), a tool for automatically evaluating text summarization, allows us to efficiently compare the effectiveness of Gist with state-of-the-art text summarization algorithms, on large text datasets. Rouge-1, a 1-gram method, with stop words, and without synonyms, is applied to compare the similarity of automatically generated summaries with expert, gold-standard summaries. Rouge-1 is shown to match human evaluations with high accuracy (Lin 2004).

Three datasets are benchmarked: Opinosis (Ganesan et al. 2010), containing 51 sets of reviews about hotels, cards, and various electronics, and professional summaries for each; single document DUC 2004 (http://www-nlpir.nist.gov/projects/duc/data/2004_data.html), containing 500 news articles from the AP and New York Times newswire, and professional summaries for each; and cmp-lg (http://www-nlpir.nist.gov/related_projects/tipster_summac/cmp_lg.html), containing 183 scientific papers from Association for Computational Linguistics conferences, with each corresponding abstract used as a professional summary.

Gist is compared to two state-of-the-art text summarization algorithms: TextRank (Mihalcea and Tarau 2004), a graph-based ranking model inspired by Google's PageRank, and LexRank (Erkan and Radev 2004), a method for computing sentence importance based on eigenvector centrality with an intra-sentence cosine similarity matrix. We omit text summarization ensembles like MEAD from our comparison to focus on the performance of individual algorithms. Gist is fully capable of working with text summarization algorithms like LexRank, in an ensemble.

Table 7 presents average recall, precision, and F-score for each dataset and algorithm. These stats, in the context of text summarization, can be interpreted as follows:

- Recall: How much necessary information does the summary contain?
- Precision: How little unnecessary information does the summary contain?
- F-score: How good is the summary? F-score is a combination of recall and precision.

Gist outperforms all comparison algorithms on all but the cmp-lg dataset, as indicated by the higher F-score. On the cmp-lg dataset, Gist shows comparable performance with a fraction of the runtime. Although Gist shows lower aver-

age recall, it consistently achieves high precision, leading to overall higher F-score. These results indicate that Gist shows improved ability to generate concise summaries while maintaining most of the necessary information extracted by state-of-the-art text summarization.

The modular nature of Gist allows for easy extension and improvement. A process of adding or modifying attributes and then testing performance on one or more datasets can be taken to improve the summarization ability of Gist. Rouge can be the test component of a modify-and-test improvement loop. Drawing from results presented in Table 7, the addition of attributes giving value to sentences based on text centrality, similar to LexRank, may improve the recall ability of Gist, leading to even higher performance.

11 Conclusion

Gist is the proposed powerful modular system for extractive summarization of text and reviews. Through metaheuristic search of sentences with Gist-calculated attribute values, Gist rapidly parses large amounts of text for the general option contained therein. This ability is proven with empirical evidence. By entering text from reviews for a product, the opinion is summarized in easily understandable human language that captures details not present in numeric ratings.

Sentiment analysis and TF–IDF relevance form the core attributes of each sentence in Gist. Additional attributes that provide heuristic guidance toward effective summary sentences are also presented in Sect. 6.1. By adding an attribute to sentences and assigning it a weight, Gist automatically integrates the new attribute into its summarization. This powerful mechanism allows for easy extension into new domains or improved summarization through a priori knowledge.

Our performance analysis shows that Gist scales linearly with the size of the text and the number of attributes, making it efficient for large-scale text parsing and data mining. Developers can easily adapt Gist to extract specific concepts from text while taking advantage of this fast performance. Such adaptation allows data miners to quickly adjust to trends or implement new ideas and improvements.

Our algorithm comparison in Sect. 9 proves Gist's ability to effectively summarize both reviews and articles. Finally, we present real summaries generated by Gist, TextRank, and LexRank. Summaries of various popular movies are presented in Table 8. For each movie, user reviews from metacritic.com are obtained and each summarization algorithm summarizes the combined review text. Tables 9, 10, and 11 each present a small selection of text relating to a topic or product, and a summary from each algorithm, generated from the text.

Compliance with ethical standards

Conflict of interest Author Justin Lovinger declares that he has no conflict of interest. Author Iren Valova declares that she has no conflict of interest. Author Chad Clough declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Arandjelović R, Zisserman A (2012) Three things everyone should know to improve object retrieval. In: IEEE Conference on Computer vision and pattern recognition (CVPR), 2012. IEEE, pp 2911–2918

Arcuri A, Briand L (2014) A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Softw Test Verif Reliab 24(3):219–250

Carenini G, Cheung JCK, Pauls A (2013) Multidocument summarization of evaluative text. Comput Intell. 29(4):545–576

Chua FCT, Asur S (2013) Automatic summarization of events from social media. In: ICWSM

Chum O, Philbin J, Zisserman A (2008) Near duplicate image detection: min-Hash and TF–IDF weighting. In: BMVC, vol 810, pp 812–815

Contour2D by MHz'as - Contour2D.jpg. Licensed under CC BY-SA 3.0 via Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Contour2D.svg#/media/File:Contour2D.svg

Deb K, Pratap A, Agarwal S, Meyarivan TAMT (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Debole F, Sebastiani F (2005) An analysis of the relative hardness of Reuters-21578 subsets. J Am Soc Inf Sci Technol 56(6):584–596

DUC 2004 PAST DATA. Document Understanding Conferences - Past Data. NIST, 2004. Web. 30 Mar 2017. http://www-nlpir.nist.gov/projects/duc/data/2004_data.html

Erkan G, Radev DR (2004) Lexrank: graph-based lexical centrality as salience in text summarization. J Artif Intell Res 22:457–479

Fiszman M, Rindflesch TC, Kilicoglu H (2004) Abstraction summarization for managing the biomedical research literature. In: Proceedings of the HLT-NAACL workshop on computational lexical semantics, Association for Computational Linguistics, pp 76–83

Ganesan K, Zhai CX, 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. Association for Computational Linguistics, pp 340–348

Gerani S, Mehdad Y, Carenini G, Ng RT, Nejat B (2014) Abstractive summarization of product reviews using discourse structure. In: EMNLP, pp 1602–1613

Goldstein J, Mittal V, Carbonell J, Kantrowitz M (2000) Multi-document summarization by sentence extraction. In: Proceedings of the 2000 NAACL-ANLPWorkshop on automatic summarization, Vol 4. Association for Computational Linguistics, pp 40–48

González-Ibánez R, Muresan S, Wacholder N (2011) Identifying sarcasm in Twitter: a closer look. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies: short papers, vol 2. Association for Computational Linguistics, pp 581–586

Hahn U, Mani I (2000) The challenges of automatic summarization. Computer 33(11):29–36

Hu M, Liu B (2004) Mining and summarizing customer reviews. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 168–177

Iversen GR, Norpoth H (1987) Analysis of variance, vol 1. Sage, Beverly Hills

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks IV, pp 1942–1948

Ku L-W, Liang Y-T, Chen H-H (2006) Opinion extraction, summarization and tracking in news and blog corpora. In: AAAI spring symposium: computational approaches to analyzing weblogs, vol 100107

Lewis DD (1997) Reuters-21578 text categorization test collection, distribution 1.0. http://kdd.ics.uci.edu/databases/reuters21578

Li F, Han C, Huang M, Zhu X, Xia Y-J, Zhang S, Yu H (2010) Structure-aware review mining and summarization. In: Proceedings of the 23rd international conference on computational linguistics. Association for Computational Linguistics, pp 653–661

Lin C-Y (2004) Rouge: a package for automatic evaluation of summaries. In: Text summarization branches out: proceedings of the ACL-04 workshop, vol 8

Loria S (2014) TextBlob: simplified text processing. TextBlob. Np

Luhn HP (1957) A statistical approach to mechanized encoding and searching of literary information. IBM J Res Dev 1(4):309–317

Maynard D, Tablan V, Cunningham H, Ursu C, Saggion H, Bontcheva K, Wilks Y (2002) Architectural elements of language engineering robustness. Nat Lang Eng 8(2–3):257–274

Mihalcea R, Tarau P (2004) TextRank: bringing order into texts. Association for Computational Linguistics Stroudsburg, Pennsylvania

Miller GA (1995) WordNet: a lexical database for English. Commun ACM 38(11):39–41

Nenkova A, McKeown K (2012) A survey of text summarization techniques. In: Aggarwal CC, Zhai CX (eds) Mining text data. Springer, Boston, pp 43–76

Nguyen P, Mahajan M, Zweig G (2007) Summarization of multiple user reviews in the restaurant domain. Microsoft Research, Redmond, WA, Citeseer

Pang B, Lee L (2004) A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the 42nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, p 271

Pang B, Lee L (2008) Opinion mining and sentiment analysis. Found Trends Inf Retr 2(1–2):1–135

Radev DR, Allison T, Blair-Goldensohn S, Blitzer J, Celebi A, Dimitrov S, Drabek E et al (2004) MEAD-A platform for multidocument multilingual text summarization. In: LREC

Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248

Rastkar S, Murphy GC, Murray G (2014) Automatic summarization of bug reports. IEEE Trans Softw Eng 40(4):366–380

Rice J (2006) Mathematical statistics and data analysis. Nelson Education, Scarborough

Saggion H (2008) A robust and adaptable summarization tool. Trait Autom Lang 49(2) 103–125

Saggion H (2014) Creating summarization systems with SUMMA. In: LREC, pp 4157–4163

Saggion H, Poibeau T (2013) Automatic text summarization: past, present and future. In: Multi-source, multilingual information extraction and summarization. Springer, Berlin, pp 3–21

Sawilowsky SS, Blair RC (1992) A more realistic look at the robustness and Type II error properties of the t test to departures from population normality. Psychol Bull 111(2):352

Sparck Jones K (1972) A statistical interpretation of term specificity and its application in retrieval. J Doc 28(1):11–21

TIPSTER Text Summarization Evaluation Conference (SUMMAC) Computation and Language (cmp-lg) Corpus. The MITRE Corporation and the University of Edinburgh, 21 May 2003. Web. 30 Mar 2017. http://www-nlpir.nist.gov/related_projects/tipster_summac/cmp_lg.html

Turney PD (2000) Learning algorithms for keyphrase extraction. Inf Retr 2(4):303–336

Uğuz H (2011) A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. Knowl Based Syst 24(7):1024–1032

Wilson T, Wiebe J, Hoffmann P (2005) Recognizing contextual polarity in phrase-level sentiment analysis. In: Proceedings of the conference on human language technology and empirical methods in natural language processing. Association for Computational Linguistics, pp 347–354

Yang J, Honavar V (1998) Feature subset selection using a genetic algorithm. In: Feature extraction, construction and selection. Springer, New York, pp 117–136

Yi J, Nasukawa T, Bunescu R, Niblack W (2003) Sentiment analyzer: extracting sentiments about a given topic using natural language processing techniques. In: Third IEEE international conference on data mining, 2003. ICDM 2003. IEEE, pp 427–434

Zhang W, Yoshida T, Tang X (2011) A comparative study of TF* IDF, LSI and multi-words for text classification. Expert Syst Appl 38(3):2758–2765