Knowledge-Based Systems 213 (2021) 106658

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A hybrid deep learning architecture for opinion-oriented multi-document summarization based on multi-feature fusion

Asad Abdi [a,b,*], Shafaatunnur Hasan [b], Siti Mariyam Shamsuddin [b], Norisma Idris [c], Jalil Piran [d]

[a] Department of Industrial Engineering and Business Information Systems, University of Twente, Netherlands
[b] School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia (UTM), Johor, Malaysia
[c] Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia
[d] Department of Computer Science and Engineering, Sejong University, Seoul, South Korea

ARTICLE INFO

Article history:
Received 20 May 2020
Received in revised form 4 December 2020
Accepted 7 December 2020
Available online 10 December 2020

Keywords:
Deep learning
Sentiment analysis
Opinion summarization
Linguistic knowledge
Recurrent neural network

ABSTRACT

Opinion summarization is a process to produce concise summaries from a large number of opinionated texts. In this paper, we present a novel deep-learning-based method for the generic opinion-oriented extractive summarization of multi-documents (also known as *RDLS*). The method comprises *sentiment analysis embedding space (*SAS*)*, text summarization embedding spaces (TSS) and opinion summarizer module (OSM). SAS employs recurrent neural network (RNN) which is composed by long short-term memory (LSTM) to take advantage of sequential processing and overcome several flaws in traditional methods, where order and information about a word have vanished. Furthermore, it uses sentiment knowledge, sentiment shifter rules and multiple strategies to overcome the existing drawbacks. TSS exploits multiple sources of statistical and linguistic knowledge features to augment word-level embedding and extract a proper set of sentences from multiple documents. TSS also uses the Restricted Boltzmann Machine algorithm to enhance and optimize those features and improve resultant accuracy without losing any important information. OSM consists of two phases: *sentence classification* and *sentence selection* which work together to produce a useful summary. Experiment results show that RDLS outperforms other existing methods. Moreover, the ensemble of statistical and linguistic knowledge, sentiment knowledge, sentiment shifter rules and word-embedding model allows RLDS to achieve significant accuracy.

1. Introduction

In recent years, with the rapid development of social media, a vast amount of reviews, opinions and feedback are constantly produced from all over the world every day. Many organizations and people follow user opinions and comment to decide the quality and performance of a product or service [1,2]. Therefore, there is a need to analyze these data to extract and detect relevant information and the polarized opinion, respectively. On the other hand, it is important for companies and people to automatically identify user's opinion whether it is positive or negative. Opinion or sentiment analysis is a technique that can be used to determine and classify people's opinion according to their polarity. Sentiment analysis is a task to find subjective information of a text such as opinions, sentiments, evaluations, etc. automatically [3–5].

1.1. Motivation

In the present time, due to the tremendous information available electronically, we need new technology or mechanism to: (i) tackle the overloading of information; (ii) obtain the information fast and efficiently; (iii) extract the most relevant and vital information; and (iv) sift vast volumes of information. Text summarization technique can be considered as a mechanism to tackle the aforementioned problems. It also helps users to quickly find the required information.

Text summarization is a process to produce a compressed version of a given source text that contains useful information for a particular user and task [6,7]. Furthermore, it is an approach to reduce the amount of textual document and present the main purpose of the source document. There are different types of summary such as a single document, multiple documents, generic, query-based, opinion-based, etc. [8,9]. In the past few

* Corresponding author at: Department of Industrial Engineering and Business Information Systems, University of Twente, Netherlands.
E-mail addresses: s.abdiesfandani@utwente.nl (A. Abdi), shafaatunnur@utm.my (S. Hasan), mariyam@utm.my (S.M. Shamsuddin), norisma@um.edu.my (N. Idris), piran@sejong.ac.kr (J. Piran).

https://doi.org/10.1016/j.knosys.2020.106658

years, a huge amount of reviews makes it extremely difficult for users to read and analyze public opinions. Thus, it is necessary to mine and summarize the opinions within large volumes of opinionated text into an easily understandable form. In other words, an opinion summarization (OS) model must be able to identify the representative opinion sentence and generate summaries with meaningful semantics. An OS is not only helpful for people but also for companies to examine and assess the opinions of costumers which can assist companies during the decision-making process. Opinion summarization is a subtask of sentiment analysis which aims to create an understandable summary from a large number of reviews on a specific topic [10–12]. An opinion summarization system includes two main steps: *sentiment analysis* and *summary generation*. The main task of summarization is to state the significant information from a source text in a concise version by retaining the main purpose and meaning of the original text, while the sentiment analysis is performed to classify the polarity of a given text, whether it is positive, negative or neutral. The systems which are based on the opinion summarization will give readers significant information about different topics. Furthermore, given the sentences related to the major subject, an automatic opinion summarizer aims to detect the polarity of sentences, and finally, summarize the positive and negative sentences.

Sentiment analysis and text summarization are the essential tasks for Natural Language Processing (NLP) with many applications such as web mining, text mining and data mining. The goal of NLP is to process text using computational linguistics, text analysis, machine learning, statistical and linguistic knowledge to extract significant information [13]. In more recent years, the use of Deep Learning (DL) model has attracted the attention of many researchers, due to its remarkable results for various NLP tasks. DL is part of machine learning architecture with multiple layers of perceptron inspired by the human brain [14]. In other words, DL is a model that contains many layers of nonlinear information processing and a method for learning feature representations at successive layers. There are various DL models such as Deep Neural Networks (DNN), Convolution Neural Networks (CNN), deep Restricted Boltzmann Machine (RBM), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and etc.

1.2. Problem statement

Most of the existing deep learning-based systems are either widely used in sentiment analysis tasks or applied to an automatic summary generation. Unfortunately, other subfields of opinion mining, such as opinion summarization, have not been studied much [15]. Hence, an automatic system that can be able to identify subjective information, classify people's opinions and summarize reviews, is required by users.

In this paper, we present a novel extractive-based sentiment summarization of multi-documents based on deep learning model to produce an opinion summary. The RDLS is based on the two pre-trained neural network algorithms which are RNN–LSTM and RBM. Each neural network algorithm is trained separately, and then the representation extracted from their hidden layers are concatenated as an input of a fusion neural network. The method is broken down into three parts which are: (i) sentiment analysis embedding space (SAS); (ii) text summarization embedding space; and (iii) opinion summarizer module. To the best of our knowledge, a deep learning-based method in which word embedding, sentiment knowledge, sentiment shifter rules, statistical and linguistic knowledge for a sentiment-oriented summarization has not been thoroughly studied. Detailed discussion of the parts of the method are as follow:

Sentiment analysis embedding space (SAS) — SAS employs a deep learning-based method, RNN–LSTM, for sentiment analysis at the sentence level. It takes the word embedding, sentiment and linguistics knowledge features as the input of RNN–LSTM. Consequently, the encoded feature from RNN–LSTM is considered as the sentence level representation. SAS aims to address the following limitations which can be summarized as follows:

(1) Currently, most deep learning approaches in NLP exploit word embedding learning algorithm [16] as a method for vector representation of each word or sentence. They usually use the freely available word2vec[1] vector that was trained on 100 billion words from Google News. One of the important drawbacks of word embedding model is that it ignores the sentiment polarity of the words [17]. As a result, words with opposite polarity are mapped into close vectors. For instance, the words "*like*" and "*dislike*" can appear in a similar context such as "*I like the movie*" and "*I dislike the movie*". With respect to their syntactic structure and word co-occurrences, both words have similar vector representations. From a sentiment point of view, their vector representation are different as they are of opposite polarities. Thus, a vector representation based on the word embedding algorithm includes insufficient sentiment information and cannot precisely capture the overall sentiment of a sentence. To overcome the aforementioned problem, the prior sentiment knowledge information is incorporated into the word embedding as word representation. Prior sentiment knowledge conveys complementary information that is not available in word co-occurrence and, hence, can enrich word embedding for sentiment analysis. In sentiment analysis, sentiment lexicons are valuable resources that can provide prior knowledge. A sentiment lexicon includes a sentiment score of a word based on its positivity and negativity that can distinguish the sentiment polarity of a word (*e.g., General Inquirer (GI)* [18], AFINN [19]). The current stage also combined several sentiment dictionaries to tackle the word coverage limit. On the other hand, various sentiment dictionaries complement each other.

(2) The word embedding model cannot differentiate the senses of a word and creates a single representation per word form. For example, the word vector for *"apple"* as fruit is equal to its word vector as a company.

(3) Contextual polarity is one of the most challenging tasks in sentiment analysis. It refers to context-based sentiment analysis, where the word's prior polarity changes with respect to a different context. The contextual issue indicates the problem of negations which may appear in different places in a sentence. If we consider an example, "*I do not like this movie*", where the negation word, "*do not*" changes the polarity of the word "*Like*". In addition, specific particles or conjunction words such as "*but*", "*despite*", etc., can affect the sentiment analysis outcome. As an example, given the sentence "*the car is good-looking but very expensive*". The but-clause changes the polarity of the previous phrase "*the car is good-looking*".

(4) On the other hand, since the performance of a sentiment analysis method relies on different types of sentences (*e.g., subjective sentences, comparative sentences, conditional/question sentences, etc.*) [20] and most of the existing methods ignore different sentence types, we consider various types of sentences in sentence-level sentiment analysis. Hence, we integrate several strategies to tackle the above-mentioned problems: contextual polarity and types of sentences.

Text summarization embedding space —To extract the most relevant information and significant part of the text, we need to determine those features that are used to identify the important information and improve the quality of the summary. To do this, TSS implements RBM algorithm with statistical and linguistic knowledge features in addition to the word embedding feature to enhance the feature vectors and extract sentence-level features.

Opinion summarizer module — This module includes two main phases which are (i) *Sentence classification,* and *(ii) Sentence selection*. Sentence classification is a fully connected neural network to

classify sentences (*i.e., "opinion summary sentence"* or "*otherwise")* using SoftMax function. To do so, the extracted sentence-level features from RNN–LSTM and RBM are concatenated to form the final extracted feature vector. Finally, the feature vector is fed into a SoftMax classifier. *Sentence selection* is used to select significant information and remove redundant information using a greedy algorithm.

1.3. Contributions

In summary, the contributions of the present work can be summarized as follows:

(1) Our work is the first attempt of a deep-learning-based method for the generic opinion-oriented extractive summarization task. On the other hand, in the context of NLP, the pre-trained models are barely used. Hence, the use of pre-trained models for feature extraction is also a major contribution to this work.

(2) To the best of our knowledge, this is the first work that combined RBM and RNN–LSTM to predict sentiment polarity and select a significant sentence.

(3) We also integrate multiple strategies to handle a few issues: (i) types of sentences; (ii) contextual polarity; (iii) word sense variations; (iv) sentiment shifter rules; (v) integration of sentiment information and word embedding; (vi) sentiment score calculation; and *(vii)* word order information and semantic relationships between words, which enable our method to achieve superior performance.

(4) The method encodes several valuable resource-information latent in a sentence to (i) generate augmented vector; (ii) learn a better sentence representation; (iii) improve the method performance. A hybrid vector is constructed for the representation of each sentence using the sentiment-based, word embedding-based, statistical and linguistic knowledge-based feature vectors.

(5) The method integrates several sentiment lexicons to tackle the word coverage limit. On the other hand, various sentiment dictionaries complement each other.

(6) The method also considers redundant sentences during the sentence selection process to increase the quality of the summary.

(7) Finally, we conducted extensive experiments to show the effectiveness of our proposed method. We report our results on the benchmark dataset: the DUC 2001, 2002 multi-document summarization datasets and the IMDB reviews. As compared to the existing methods, the method achieves superior performance on all the measure metrics and the significant results affirm the suitability of the proposed method for opinion-oriented summarization. In addition, a combined architecture of RBM and RNN–LSTM model performs better than RBM and RNN–LSTM models alone to produce an opinion summary.

1.4. Paper organization

The rest of the paper is structured as follows. We summarize the recent related works in the next section. Our proposed method, RLDS, is discussed in detail in Section 3. Our experimental results are explained in Section 4. Finally, we provide conclusions and suggest some ideas for future research.

2. Related work

Sentiment analysis — with the huge amount of user-generated texts, extraction of significant information from numerous documents has gained much attention from the community of NLP [21]. Sentiment analysis is an active research area of NLP that aims to identify subjective information and determine the sentiment orientation (*e.g., positive or negative*) of a given text [22]. In other words, sentiment analysis is the computational study of people's opinions, which aids users to gather desirable information for decision making. Recently, sentiment analysis has been applied to various domains such as commercial (e.g., market prediction, mining reviews, political, social media [23–25]. In general, sentiment analysis can be divided into three levels [26]: (i) document level; (ii) sentence level; and (iii) aspect level. In the document level, the sentiment analysis process determines the overall sentiment polarities of a given document. It assumes that the document discusses only one topic or a single entity. Similar to document-level sentiment analysis, the task of the sentence level is to determine whether a sentence expresses a positive, negative or neutral opinion. This level also examines the subjectivity classification, which aims to determine whether a sentence is an objective (*not opinionated*) or subjective (*opinionated*). As compared to document and sentence level sentiment analysis, aspect level sentiment analysis is employed to acquire details of the opinionated text. In other words, it extracts the people's opinion expressed on entity and feature/attribute of the entity. The task of aspect-based level sentiment analysis includes three main steps: entity/object identification, feature/attribute extraction and attribute polarity identification. Currently, most sentiment analysis method can be divided into three categories which are lexicon-based approach, machine learning approach and hybrid approach [15]. A dictionary-based method determines the polarity of a text-based on the total sum of comprised positive or negative sentiment words in a text (e.g., (Balahur et al. 2012); (Yadav and Chatterjee 2016)). A sentiment lexicon includes a set of negative and positive values assigned to corresponding words and phrases (e.g., AFINN (Nielsen, 2011), Sentiment140 Lexicon (Mohammad, Kiritchenko, & Zhu, 2013)). In comparison with machine learning-based method, the lexicon-based method is suitable for sentiment analysis, since it needs fewer resources and does not need some annotated corpora. Furthermore, a sentiment lexicon can also be used in a machine learning-based approach in order to make sentiment related features. A lexicon-based approach suffers from word coverage of sentiment words in a text. As compared to the lexicon-based approach, a machine learning-based approach uses the most common machine learning algorithms to classify sentiment orientation. In this type of method, feature definition/extraction is very important for a learning-based method. Many studies used the most common standard classifier for sentiment analysis such as Support Vector Machine (SVM) (e.g., [27,28]) or Naive Bayes (e.g., [29]. Finally, in a hybrid approach, the dictionary-based methods were combined with machine learning-based approaches (e.g., [30]) to improve the accuracy of a text classification system. Recently, deep learning has also been applied to sentiment analysis and achieved promising results [15,31].

Deep learning is a neural network (NN) with multiple hidden layers. A neural network contains various number of nodes per layer and hidden layers between the input and output layers. Given an input, a NN approach is able to learn features and to produce a classification result. Some researchers found that the traditional sentiment analysis approaches are insufficient, and the results are not satisfied [32]. Hence, a deep learning model has achieved significant attention in various NLP tasks. Several deep learning-based methods have been proposed for sentiment analysis tasks such as studies by Caliskan, Bryson and Narayanan [33] and Liu, Liu, Shan and Wang [34].

There are two types of opinions: (i) regular opinion that can be categorized into direct and indirect opinion wherein direct opinion, a user expresses his/her opinion/feeling directly on an attribute, while in indirect opinion, an opinion is expressed indirectly on an attribute; and (ii) comparative opinion [35] which expresses the difference between entities. In addition, a review text can be an explicit review or implicit review text. An explicit

review is a subjective representation that demonstrates regular or comparative opinion, while an implicit review is an objective representation that indicates regular or comparative sentiments. A subjective text expresses subjective views and feelings, while an objective text expresses factual information, no sentiment or opinion [36]. Emotion is also related to sentiment analysis that presents our feeling and opinion. The researches categorized people's opinions into some groups such as, "*love*", "*joy*", "*surprise*", "*anger*", "*sadness*", and "*fear*".

Summarization — text summarization can be considered as a technology to identify the most important information. The main task of summarization is the process of producing a short version of a given text document that provides the most important information and retains the main meaning of the source text [37,38]. It also solves the problem of selecting the most important fragments of a source text. Text summarization is an active field of research in NLP area that can be used for several NLP tasks such as question answering, text classification, or information retrieval.

In general, the process of text summarization can be classified into three main stages [39]: (i) *analysis stage*, how to select a few significant features; (ii) *selection stage*, how to select a subset of ranked units (e.g., words, phrases or sentences); finally, (iii) *the generation stage* that creates an appropriate summary using the selection stage.

Text summarization methods are classified into two categories: extractive and abstractive summarization [40–42]. An extractive summarization method is used for selecting a representative subset of the sentences, paragraphs from the source text to produce a summary. On the other hand, an abstractive summarization method needs a deep understanding of the concepts presented in the original text. It tries to understand the original text and then presents it in a concise form with new concept, which demonstrates the most significant information from the original text. Besides these facts, there are also two groups of text summarization which are indicative and informative. An indicative summary gives the main idea of the original text, while an informative summary gives brief information of the original text. A text summarization method can also be performed on a single document, which deals with one document, or multi-document, which produces summary from a collection of related documents [43]. Similarly, a summarization system aims to generate a generic summary, where it considers the total document, or to summarize the content of a document that is most relevant to a user query. Several supervised, unsupervised and deep learning-based methods have been proposed for text summarization. Starting with unsupervised methods, Maximum Marginal Relevance (MMR) [44] is one of the well-known approaches that is used for text summarization. It employs a greedy method to select sentences and considers the redundancy. On the other hand, the graph-based model plays a key role in the text summarization area, due to its ability to reflect various sentence relationships. Therefore, a variety of methods based on the graph-based model have also been proposed for text summarization such as [45,46]. In contrast to the unsupervised method, there are also several machine learning-based summarization approaches which used a combination of appropriate features and learning algorithms. This type of model learns how to generalize its parameters to extract significant passages [47]. Recently deep learning approaches have also been used for abstractive and extractive summarization. Zhong, Liu, Li and Long [48] proposed a method based on the auto-encoder (AE) algorithm for extractive summarization. Denil, Demiraj and de Freitas [49] developed a model based on the CNN algorithm to output significant sentence to be included in the summary. Duraiswamy [50] introduced a method to produce a generic summary using RBM algorithm.

Opinion summarization —recently, there is a huge amount of reviews, therefore, reading and analyzing all these reviews is a big problem and is not feasible. An opinion summarization aims to tackle this problem, extract the most relevant information and produce an understandable summary from an opinionated document [51]. In other words, the opinion summarization system takes as input a large number of reviews. Subsequently, it processes all the given reviews and produces a summary of all the input opinionated text. With opinion summaries, users have the best information to make better decisions [52]. An opinion summarization can be considered as multi-document summarization although, they are different from each other. The opinion summarization summarizes opinions from a large number of the opinionated text while a traditional summarization focuses on extracting informative information and removing redundancy [53]. Consequently, text summarization and sentiment analysis must be integrated to generate an opinion summary. Text summarization identifies most relevant sentences from a review text and the sentiment analysis component determines and classifies objective or subjective sentences and their polarities (positive, negative or neutral), respectively [26]. Although there are several works on text summarization (e.g., [54]) and sentiment analysis (e.g., [55]), there are only a few works on the combination of these two areas [56].

Yadav and Chatterjee [57] proposed an approach to select important opinionated sentences of a document. The approach follows the following processes: First, it applies the stop word removal, sentence splitting, porter stemming algorithm and part-of-speech tagging on a dataset. Consequently, the approach assigns a sentiment score to each word according to its POS tag using the SentiWordNet database. Finally, the sentence sentiment score equals the sum of the sentiment scores of all the words in the sentence. [58] introduced an approach (OM+ Summarizer) to produce opinion summaries. The proposed method performs two main tasks: (i) the first step determines the opinionated sentences and their polarities. Subsequently, the subjective sentences are sent to (ii) the summarizer (LSA-based [59] text summarization) to produce a summary. Kim and Calvo [60] proposed a system based on the sentiment score-based technique and a lexical resource to identify and extract subjective sentences to create a sentiment summary. To do this, the system identifies important words using the TF–IDF approach, then assign a sentiment score to each word using the SentiWordNet 3.0. In the end, the sentences are ranked based on their scores (*sentence score= sum of the words sentiment score in a sentence*), and the highest-ranked sentences are selected to create the summary. Kabadjov, Balahur and Boldrini [61] proposed a method to summarize opinions expressed in texts. It includes two main steps: (i) Sentiment Analysis step identifies the subjective sentences, then assigns to each sentence its polarity (positive and negative sentences) and sentiment score using three different lexicons (*e.g., WordNet-Affect, SentiWordNet, MicroWNOp*); (ii) Summarization Algorithm which consists of two main tasks, rank all sentences using their sentiment score and select highest *n* sentences.

3. Proposed method

In this paper, we present a method for extractive-based opinion summarization which consists of two levels as shown in Fig. 1. The first level involves the *pre-processing module*, *sentiment analysis embedding* space (SAS) and *text summarization embedding space* (TSS). The second level consists of *opinion summarizer module* (OSM). The idea is that the neural network technique for each input space of the first level (*SAS, TSS*) is trained on corresponding benchmark dataset independently. The vector representations extracted from their hidden layers are then concatenated as an input of a classifier (*OSM at the second level*) which takes advantage of cross-space features drawn from the sentence analysis embedding space and text summarization embedding space.

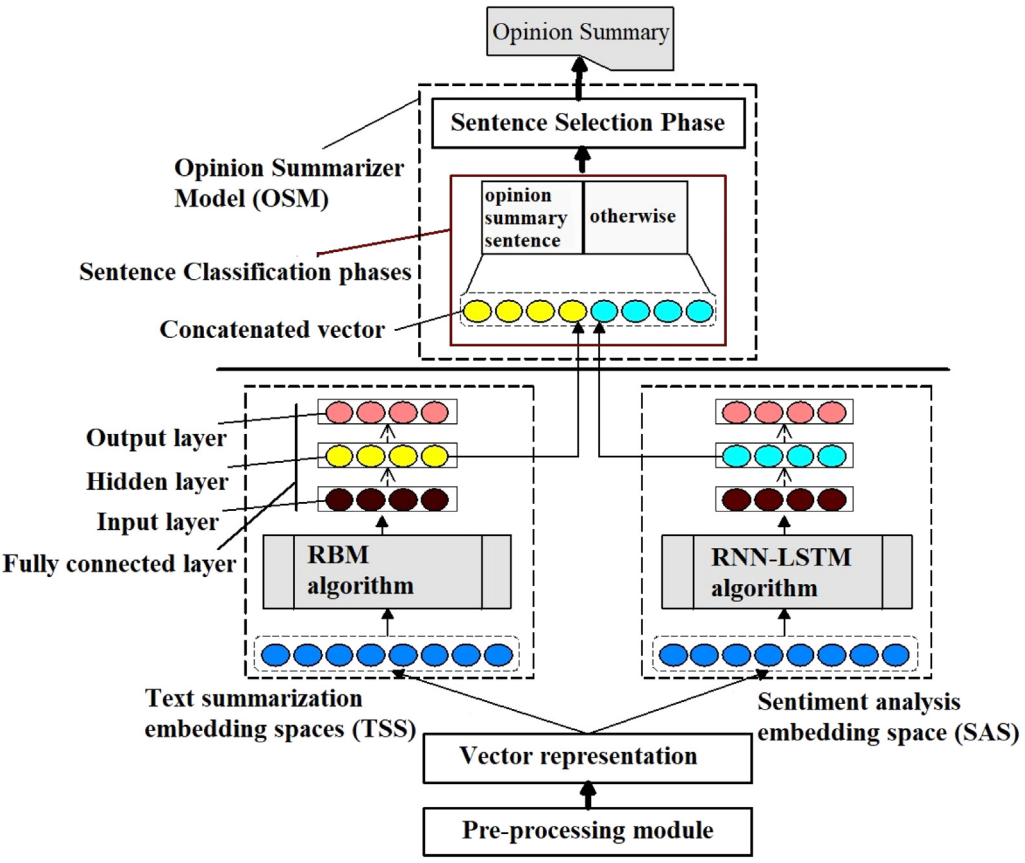

Fig. 1. The architecture of the proposed method RDLS.

(1) Pre-processing module: it includes basic linguistic functions. The output of the pre-processing will be the set of hand-crafted features and word embedding representation.

(2) Sentiment analysis embedding space: it employs RNN–LSTM algorithm, on top of pre-trained word vectors and hand-crafted features (*sentiment and linguistics knowledge features*) for the sentence-level classification task. We trained SAS on a sentiment benchmark dataset, where this pre-trained model is used for sentiment feature extraction.

(3) For the *text summarization embedding space*, we trained the RBM algorithm that starts from word embedding and various features (*statistical and linguistics knowledge features*) extracted from corresponding dataset. The pre-trained RBM algorithm is employed to enhance those features, produce a certain set of features for each sentence and improve sentence selection to compose the summary.

(4) The second level concatenates the representations extracted from the hidden layers of subsystems (TSS, SAS) to form the final feature vector. The final feature vector from the previous step is fed as input to the fully connected network (OSM) to classify sentences (*e.g., opinion summary sentence; otherwise*). Finally, the sentence selection phase is employed to generate an opinion summary.

A detailed description of these levels is provided in the following subsections.

3.1. Pre-processing

Before applying RDLS method, we need to perform data pre-processing to produce a higher quality of opinion summary and reduce the computational complexity. Briefly, the steps for pre-processing are as follows:

The raw text is first given as an input to the pre-processing step. In this step, each document is decomposed into several paragraphs. Meantime, the track of each paragraph, each sentence of the corresponding paragraph and the position of each sentence in its respective paragraph are kept. Subsequently, the paragraphs are further decomposed into sentences. In the next step, tokenization, a basic approach of text pre-processing, splits the sentences into words. The generated tokens (*words*) are used for various purposes such as to identify term frequency, key-words, inverse document frequency, etc. The sequence of tokens is also passed to the next procedure, stop-word removal, to eliminate words which do not convey any meaning and are useless for text mining. The stop-word[1] includes words like articles, conjunctions, prepositions, and pronouns which are common words with little useful information. We also employ the stemming procedure to reduce a word to its root form. Part-of-speech (POS) tagging is also used to classify the words of text on the basis of part of speech categories (e.g., noun, verbs, adverb, adjectives) they belong. The POS tagging provides useful lexical information.

Sentiment shifter rules — the pre-processing step also applies sentiment shifter rules on datasets. The sentiment shifter refers to the context-based sentiment analysis. It includes some words like *"but", "while", etc.* that can change the sentiment orientation of the sentence following them [62]. Consider an instance, "*the car is good-looking but very expensive*", where the word "*but*" changes the polarity of the sentence "*the car is good-looking*". In other words, the polarity of sub-sentence before the word *"but"* and after it is opposite to each other. Therefore, the overall opinion is only in the sub-sentence following the word "*but*". Hence, the

[1] http://xpo6.com/list-of-english-stop-words/.

Table 1
Sentiment shifter rules for text sentiment analysis.

Example	Semantic rules
"I am sorry $_{sub-sentence1}$, but I can't pay you $_{sub-sentence2}$"	If a sentence includes "*but*", ignore the sentiment orientation of sub-sentence$_1$ and only consider the sentiment of the sub-sentence$_2$.
"He played well $_{sub-sentence1}$, despite being injured $_{sub-sentence2}$"	If a sentence includes "*despite*", only consider the sentiment of the sub-sentence$_2$.
"I wouldn't wear those shoes $_{mainclause}$ unless I was trying to break my ankle $_{subordinateclause}$"	If a sentence includes "*unless*", and the "*unless*" is followed by a negative clause, ignore the "unless" clause.
"My throat is killing me, and while I got a decent night's sleep last night, I still feel like I'm about to fall over"	If a sentence includes "*while*", ignore the sentence following the "*while*" and consider the sentiment only of the sentence that follows the one after the "*while*"
"I am glad I got the job; however, I'll have to travel more often"	If a sentence includes "*however*", ignore the sentence before "*however*" and consider the sentiment of the sentence after "*however*"

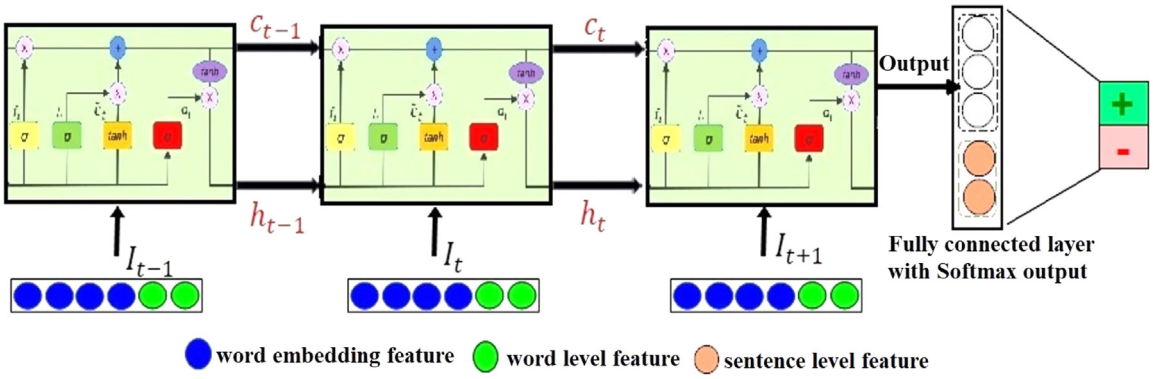

Fig. 2. The architecture of sentiment analysis embedding space (SAS).

sub-sentence before the word *"but"* is unessential part and can be removed. The aforementioned pitfall leads to the potential misclassification in sentiment analysis. Thus, a set of rule strategy is required to deal with the specific particles (*e.g., "but", "while", 'etc.*) that can affect the result of sentiment analysis. In this work, we used the rules proposed by [63,64], as shown in Table 1.

Sentence vector representation — a popular approach for text representation is bag-of-word (BOW) approach. In this approach, a sentence is represented using a set of its words where each word is weighted using various methods (*i.e., term frequency (TF), True/False, 0/1, term frequency–inverse document frequency (TF–IDF)*). BOW suffers two main issues [65]: (i) a vector representation based on BOW is sparse since each sentence only includes a small number of words.

A sparse representation can cause a problem for the training process since some words may only appear in the testing dataset and never be seen in the training dataset. (ii) BOW is not able to distinguish the meaning of two sentences because it ignores the syntactic structure and word order in sentences. In other words, different sentences can have the same vector representations. The *n*-gram is another popular method that performs best [66]. It considers the word order in a sentence, but it also suffers from data sparsity and high dimensionality.

To tackle the aforementioned problems, recently most of the proposed systems used the word embedding technique to produce a low-dimensional vector for sentence representation. Due to its ability to encode syntactic and semantic properties of words, the word embedding technique has been used for various NLP tasks such as parsing [67], POS tagging [68], etc. Word2vec[2] is one of the popular word embedding models which is used to perform the computation of the word vector representations. It includes Continuous Bag-of-Words model (CBOW) [69], and Skip-Gram model (SG) [16] to provide high-quality word embedding vectors. The CBOW predicts the target word based on the embedding of its context words, while the SG predicts the surrounding words given the target word. We also employ the word2vec approach in our proposed method and consider several strategies to address its drawbacks.

3.2. Sentiment analysis embedding space (SAS)

As shown in Fig. 2, the SAS architecture comprises the following parts: input vectors, RNN–LSTM layer, concatenation layer and fully connected layer with SoftMax output. In input vectors, sentences are firstly considered via the pre-processing step to standardize the sentences and capture only important information containing the sentiment of the sentence. Therefore, the integration of word-level embedding, sentiment and linguistic knowledge features (*word-level feature*) are fed into the RNN–LSTM layer to generate a sentence-wide feature set. In the concatenation layer, we augment the extracted vector representation from last LSTM cell with the sentiment and linguistic knowledge features (*sentence-level feature*) to form a final vector representation for each sentence. This vector is taken to the fully connected layer, then the SoftMax function reveals the sentiment label, subjective (*positive/negative*). We explain in detail the kinds of RNN and LSTM that we use as follows:

Some background on RNN — the RNN is a popular neural network algorithm that has been used in many NLP tasks. RNN is able to model variable-length input sequences and learn long-term dependencies among the sequence [70,71]. In other words, the ability to capture contextual information in sequential data can

[2] https://code.google.com/archive/p/word2vec/.

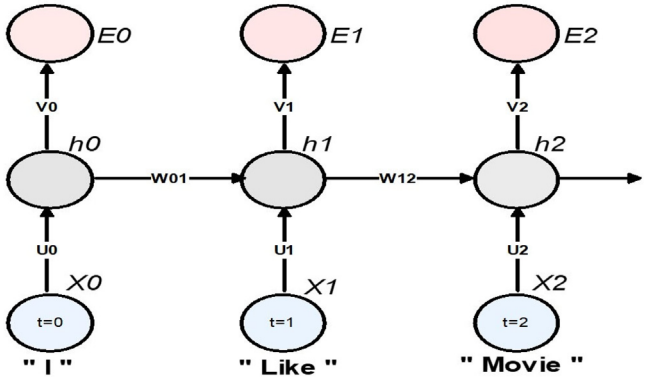

Fig. 3. The architecture of Recurrent Neural Network (RNN).

be useful to obtain semantics of long text. Fig. 3 shows a recurrent neural network architecture and the connections between the hidden units. The hidden unit of an RNN is equal to $h_t = f(X_t U_t + h_{t-1} W_{h_{t-1}h_t})$, where t is time step, W, U, V are the weight matrix and f is an active function. In an RNN model, each word (*hidden unit*) affects all the subsequent words (*hidden units*). Given the sentence "*I like movie*", the RNN processes the sentence as follows: at the first step ($t = 0$), it processes the word "*I*" and computes the hidden unit h_0 ($h_0 = f(X_0 U_0)$). A second step ($t = 1$), the word "*like*" is processed and the h_0 also fed into hidden unit $h_1 (h_1 = f(X_1 U_1 + h_0 W_{01}))$. Finally, it processes the word "*movie*" and feeds the h_1 into h_2 ($h_2 = f(X_2 U_2 + h_1 W_{12})$).

Word order information and semantic relationships between words have an important impact on sentiment classification performance. Since the RNN model deals with time-series data and can learn from a sequence of words, it can solve the problem of the negation which may appear in different places in a sentence. If a negation word (*e.g., not, never, etc.*) appears in a sentence, the hidden unit of this word will affect the polarity/ emotion of the subsequent words; and as a result, it will affect the polarity of the sentence. For instance, the words *"bad"* and *"not"* are negative, but the phrase *"not bad"* which is combined of these two words has a positive meaning.

RNN suffers from vanishing gradients [72]; hence, (i) the length of the sequences that an RNN can process is limited and (ii) RNN is not able to keep track of long term dependencies. Therefore, LSTM network was proposed to tackle the long-distance dependencies problem of RNN using a memory cell that preserves state over long periods of time [73]. The core of LSTM is a memory cell which four main gates regulate the information flow into and out of the cell: the input gate, output gate, forgetting gate and candidate memory cell (Fig. 4). The cell decides what to keep in and what information to erase from memory via gates. An LSTM architecture is not fundamentally different from RNN, but it employs different functions to calculate the hidden state. The LSTM memory cell is updated using the following steps:

working of gates — the gates are calculated at every timestep t using the following equations:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (2)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$

$$\widetilde{c_t} = Tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$

where h_{t-1} is the hidden unit at time step $t-1$, x_t is the input at time t. b is bias vector and subscript $i, f,$ o and c indicate the input gate, forget gate, output gate and cell state respectively. U and W indicate the weight matrix of each gate. i_t, f_t, o_t and $\widetilde{c_t}$ represent the input gate, forget gate, output gate and candidate memory cell state respectively. *Tanh* and σ are hyperbolic tangents and sigmoid function respectively.

Memory cell update — in this step the cell state at time t is updated using the Eq. (5). The c_{t-1} is multiplied by f_t, then the result of $(i_t \times \widetilde{c_t})$ is added.

$$c_t = f_t \times c_{t-1} + i_t \times \widetilde{c_t} \quad (5)$$

Where c denotes the variable for cell state.

Hidden layer output — given the new state of the memory cells, c_t, and o_t the hidden state, h_t can be computed as follows:

$$h_t = o_t \times Tanh(c_t) \quad (6)$$

since RNN can process sequential data and the LSTM can control the flow of information, to cope with the vanishing gradient and aid the RNN to capture long-term dependencies, we employ RNN–LSTM in the sentiment analysis process.

3.2.1. Input features

A feature is a characteristic of a text for capturing patterns in data. Feature extraction process transforms the input dataset into a set of features to improve the overall quality of text classification. The procedure of feature extraction is explained in detail as follows:

i. Word embedding

A neural network needs a vector representation of each word or sentence as an input to the network. Word embedding is a method that can be used for feature learning. It is able to transform words into real-valued and low-dimensional vectors and capture useful syntactic and semantic properties about the words. We use word2vec to translate a sentence into its vector representation. The word2vec was trained on 100 billion words from Google News. In our method, given a sentence $S = \{w_1, w_2, \ldots, w_n\}$, where n indicates the number of words in S. Each word, w_i, is associated with d-dimensional vector embedding, $X_i \in R^d$. The sentence of length n is represented as follows: All the word vector representations are concatenated in their respective order, $X_{1:n} = X_1 \oplus X_2 \oplus \cdots \oplus X_n$, where $\oplus$ is the concatenation operator. Furthermore, each sentence is padded with zero-vectors to a fixed length.

ii. Sentiment and linguistic knowledge

Although the word embedding learning method has been applied to various NLP tasks, it is not effective enough for sentiment analysis. The word embedding approach has some obvious drawbacks which can be summarized as follows:

a. Word-level features

Part-of-speech (POS) tagging — word-sense disambiguation (*when a word has multiple meanings*) is one of the problems with the word embedding approach. It cannot distinguish the senses of words and creates a single representation per word form [74]. For example, given two sentences, *"The coach devised a great play"* and *"The boy went out to play in the yard"*, where each sentence associates with a different meaning of the word "*play*" based on the context of the word's usage in a sentence. To resolve the problem, we use a simple approach, POS tagging, as an additional feature for each word: ("*The/DT children/NNS went/VBD out/RP to/TO play/VB in/IN the/DT park/NN*", "*The/DT coach/NN devised/VBD a/DT great/JJ play/NN*"). We use a constant binary vector (six-dimensional vector: "*noun*", "*verb*", "*adjective*", "*adverb*", "*preposition*", "*conjunction*") and concatenate with the corresponding word embedding vector.

Sentiment-encoded word embedding — the most serious problem with the word embedding approach is that the approach ignores

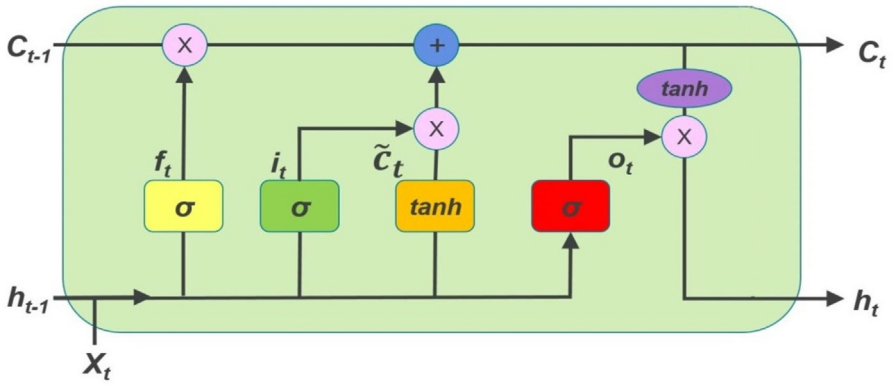

Fig. 4. The LSTM architecture.

the sentiment information of a word. As consequence, two words with similar contexts and opposite polarity (e.g., *"poor" and "nice"* in *"He is a nice guy"* and "*He is a poor guy*") are mapped into close vectors in the embedding space. From a sentiment point of view, the vector representations of both above-mentioned words should be very different as they convey opposite polarity. Given a vector representation that has insufficient sentiment information, the SAS is not able to accurately obtain the overall sentiment of the sentence, therefore, it affects the sentiment classification performance. To tackle the current problem, we encode the sentiment information into the word embedding process to learn better representation for sentiment analysis. To do this, we use a 2-dimensional binary vector of prior sentiment polarity of words (positive/negative polarity). For each word of the input sentence, the corresponding sentiment polarity is provided by the prior knowledge or external source such as sentiment dictionary. One problem that we encounter is that a word of the sentence is not in the lexicon. In such case, we employ the SSM method as described in the previous section. The binary vector is appended to the end of the corresponding word embedding. In sentiment analysis, a sentiment lexicon can provide prior knowledge. A sentiment lexicon contains a set of words and their corresponding sentiment score/polarity. In our work, we combine ten sentiment dictionaries in order to tackle the word coverage limit of an individual sentiment dictionary (*called MSD*). We explain the combination process in the following section:

Sentiment lexicons combination — we integrate several sentiment dictionaries. The existing dictionaries with different size and format are mapped to three categories as follows. Since the lexicons have various formats, we first standardize them, then, the score of each word in MSD is computed by averaging sentiment scores of overlapping words.

Category 1: a sentiment dictionary includes sentiment score with various numeric ranges (e.g., (Nielsen, 2011), [−5, +5]; [75], [−7, +7]; [76], [−1, +1]; [77], [−5, +5]; [78], [0,1]. For this type of lexicons, a sentiment score is normalized from [−5, +5] or [−7, +7] or [−1, +1]. Regarding Baccianella, Esuli and Sebastiani [78], each row of this lexicon includes a synset with POS information, an ID that maps the synset to WordNet, the positive (Pos)/negative (Neg) sentiment scores and a gloss which contains the meaning and sample usage of the terms present in the synset. Each synset is a group of terms that are synonyms of one another. A synset is "*objective*" if the following equation is equal to 1, $ObjScore = 1 - (PosScore + NegScore)$. Eq. (7) [79] is used to calculate the score of each word within the range of [−1, 1]:

$$Senti\_score = (Positive_{score} - Negative_{score}) \tag{7}$$

If *senti_score* is greater than zero, less than zero or equal zero the sentiment word orientation is positive, negative or neutral/objective, respectively.

Category 2: The sentiment dictionary classifies the words into "*positive*", "*negative*" and "*neutral*" (e.g., [80] and [36]). In this type of category, we assign +1, −1, 0 to positive, negative and neutral words, respectively.

Category 3: The sentiment dictionary classifies the words into several types of emotions such as *"bad"* and *"happy"* or categorize into several groups such as "*positiv", "affil", "strong", "weak", "fail", "passive*" (e.g., [81] and [18]). Therefore, we assign (+1) to words from "*positive-emotion" or "positive-group", (−1) to "negative-emotion" or "negative -group" and (zero/0) to "neutral-emotion".*

Semantic Sentiment Method (SSM) — as mentioned above, the limited words coverage is a major limitation of a sentiment lexicon. We exploit the SSM to specify the score of a word if it does not exist in MSD. Let $WS = \{W_1, W_2 \cdots W_N\}$ include the synonymous words of a word (W). For each W of WS, the algorithm 1 performs the following tasks: *(1)* if the W exists in the MSDL, then SSM returns its sentiment score (SC); *(2)* the SSM checks, if the SC value is greater than 0 (*zero*), then SSM adds the SC to the Pos_{sw}; *(3)* if the SC value is smaller than 0, then SSM adds the SC to the Neg_{sw}. Finally, the following equation is used to compute keywords Total Sentiment Score of W.

$$\text{Total Sentiment Score}\,(W) = \frac{\sum Pos_{sw} - \sum Neg_{sw}}{m+n} \tag{8}$$

Let n and m indicate the number of positive and negative words respectively. $\sum Pos_{sw}$ and $\sum Neg_{sw}$ are defined as follows: $\sum Pos_{sw} = \sum SC_{positive}$ and $\sum Neg_{sw} = \sum SC_{negative}$.

b. Sentence-level features

Apart from word-level features, there are sentence-level features that cannot be ignored for training the sentence-level. The following features are extracted at the sentence level:

Lexicon feature:

- The number of positive sentiment tokens in a sentence.
- The number of negative sentiment tokens in a sentence.

The total sentiment score feature:

- The sum of the scores of the sentiment tokens.

Negation features:

- Frequency of individual negation words within a sentence.

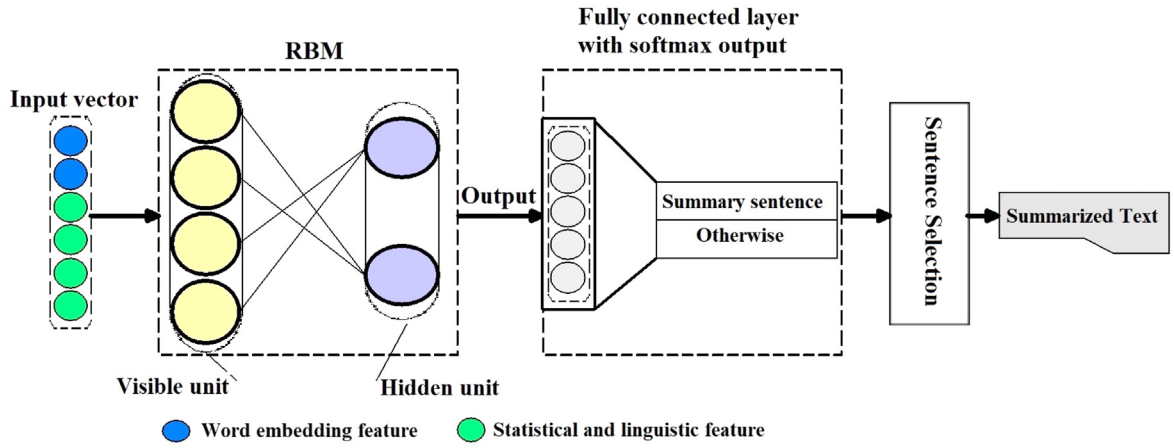

Fig. 5. The architecture of Text summarization embedding spaces (TSS).

Punctuations feature:

- Frequency of exclamation ("!") mark.
- Frequency of Question ("?") mark.

POS feature:

- Frequency of nouns, adjectives, verbs, adverb.

Sentence types:

We also considered the different types of sentences as a feature [20,26]: *(1)* subjective/ objective sentence: an objective sentence expresses factual information, while and a subjective sentence expresses an opinion or sentiment; *(2)* interrogative sentence/conditional sentence: an opinionated sentence may not present any opinion or sentiment such as, "*may you tell me which Samsung laptop is good?*", "*If I can find a good laptop in the shop, I will buy it*" and "*is your bicycle in a good condition?*". These sentences contain sentiment words, but they do not express a positive or negative opinion on the laptop. However, all conditional and question sentences do not express opinion or sentiments. We use four types of sentences (*objective, subjective, question, conditional*) encoded as a 4-dimensional binary vector.

- Is the sentence subjective?
- Is the sentence objective?
- Is the sentence question/ interrogative sentence?
- Is the sentence conditional sentence?

3.3. Text summarization embedding spaces (TSS)

The neural network model designed for TSS is based on Restricted Boltzmann Machines algorithm. The main principle of this section is to address the selection of important sentences from a text for a summary. Fig. 5 presents a general architecture of the proposed model of our deep network. The model consists of input vectors, RBM algorithm, and summary generation.

To achieve efficient representation of the text, the input sentences are converted into numerical values, called vector representation, for the correct realization of a text summarization solution. The feature extraction is done on the text obtained after the pre-processing step. Then, a concatenation of word-level embedding, statistical and linguistic knowledge features are submitted to the RBM algorithm. We also pad zeros at the beginning and end of the sentence. The RBM algorithm is used to reduce the length of input feature vectors (dimensionality reduction) in order to produce complex features out of simple ones and improve the quality of the summary. Consequently, the sentence vector is passed through the hidden layer in which each feature vector value is multiplied by learned weights and a bias value is added to all the feature vector values which are also learned by the RBM. It is worth noting, the input layer size depends on the number of input features that are extracted using the word-level embedding, statistical and linguistic knowledge. The output layer size is variable and usually represents a concise version of the input feature vector.

Summary generation is performed using two steps, *sentence classification* and *sentence selection*. Above the RBM algorithm, there is a fully connected SoftMax layer. The SoftMax classifier as the output layer of a neural network is used to predict the label for each sentence (e.g., "*summary sentence*" or "*otherwise*"). Any sentence with label "*summary sentence*" is considered as a candidate summary sentence. Once the sentences were classified by the fully connected layer into the two labels, a simple approach is used to create the final summary. To do this, *Sen_Vec* that includes a pair of the candidate summary sentence and its corresponding feature vector is passed to the "*Sentence Selection*" step to produce the final summary.

$Sen\_Vec = \{(Sen_1, Vec_1), (Sen_2, Vec_2) \cdots (Sen_n, Vec_n)\}$, where Sen_1 indicates a candidate sentence for summary text and Vec_1 demonstrates the corresponding feature vector extracted using the section "*3.3.1. Input features*"). For instance, *Vec* = (*title, sentence position, proper noun score, cue-phrases, ...*). In the sentence selection step, firstly, each sentence score is computed using the sum of its feature vector values. Let $Vec_1 = (2, 0, 3, 0, \ldots)$ be the corresponding vector of sentence Sen_1 and $Sen_{score} = (2 + 0 + 3 + 0 + \cdots)$ is the sentence score of sentence Sen_1. Secondly, the sentences are sorted in descending order based on the Sen_{score}. Then, a sentence with the highest score is added into a summary, *S*. The next sentence is added into *S*, if its Jaccard similarity [82] with other sentences of *S* does not exceed the predefined threshold. This process is repeated until a user specified summary limit is reached. At the end, the set *S* is considered as a final summary.

Restricted Boltzmann Machine or RBM[3] is an undirected graphical model based on a bipartite graph [83–86]. As shown in Fig. 6, the model includes two main layers: a visible layer (*input data*) and a single hidden layer (*feature detectors*). RBM is a variant of Boltzmann machines, with the restriction that a pair of units from the visible and hidden layers have a connection, and there is no connection between units within a layer. The connection between units is symmetric where the information flows in both directions and the weights are the same in both directions. Since our input data includes the real values, we use Gaussian–Bernoulli RBM.

[3] https://pathmind.com/wiki/restricted-boltzmann-machine.

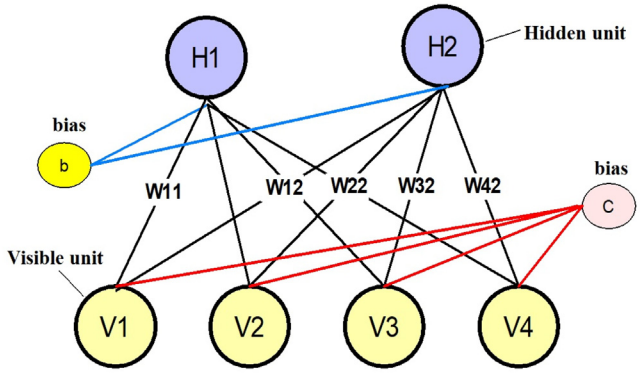

Fig. 6. The architecture of Restricted Boltzmann Machines (RBM).

The energy function of Gaussian–Bernoulli RBM is defined as follows [85,87]:

$$E(v,h|\theta) == -\sum_{i=1}^{n_v}\frac{(v_i-b_i)^2}{2\sigma_i^2}-\sum_{i=1}^{n_v}\sum_{j=1}^{n_h}w_{ij}h_j\frac{v_i}{\sigma_i}-\sum_{j=1}^{n_h}c_jh_j \quad (9)$$

where v is the real-valued input feature, h is a hidden unit. b and c are biases of hidden unit and visible unit, respectively, w_{ij} is the weights between visible units and hidden units and σ is the standard deviation of the visible unit.

The conditional probability of the visible unit and the hidden unit can be derived as follows:

$$\begin{aligned}P(v_i|h) &= \frac{1}{\sigma_i\sqrt{2\pi}}\exp\left(-\frac{1}{2\sigma_i^2}(v_i-b_i-\sigma_i\sum_j w_{ij}h_j)\right)\\ &= \aleph(v, b_i+\sigma_i\sum_j w_{ij}h_j, \sigma_i^2)\end{aligned} \quad (10)$$

$$\left(P(h_j=1|v\right) = sigm(c_i+\sum_i w_{ij}\frac{v_i}{\sigma_i}) \quad (11)$$

where $\aleph(v,\mu,\sigma^2)$ indicates a Gaussian distribution with the mean μ and the variance σ^2, and $sigm(x)=\frac{1}{1+\exp(-x)}$.

Parameters update, the parameters of RBM, w_{ij}, b_i and c_i can be updated as follows:

$$\nabla W_{ij} = \frac{1}{\sigma_i}(\langle v_ih_j\rangle_{data}-\langle v_ih_j\rangle_{recon}) \quad (12)$$

$$\nabla b_i = \frac{1}{\sigma_i^2}(\langle v_i\rangle_{data}-\langle v_j\rangle_{recon}) \quad (13)$$

$$\nabla c_i = (\langle h_i\rangle_{data}-\langle h_j\rangle_{recon}) \quad (14)$$

We denote by $\langle.\rangle_{data}$ and $\langle.\rangle_{recon}$ the expectation over the data and model distributions, respectively.

3.3.1. Input features

In this step, each sentence from the multiple documents is structured into a feature vector. To create the most suitable and informative summary, we consider various features for the computation. The feature sets are:

i. Word Embedding

Each sentence is converted into a vector representation using the word2vec model. Then, the average over the vectors of all words of the sentence will be presented as a sentence vector. Finally, the sentence vector is concatenated into the statistical and linguistic knowledge-based feature vector.

ii. Statistical and linguistic knowledge

The following significant features are extracted for all sentences.

• Title

A sentence is important if it includes words that occurred in the title or major headings of a document [88]. The score of title feature is the ratio of the number of tokens in the sentence that occur in title to the total number of tokens in the title.

• Cue-phrases

Cue-phrase includes set of words[1] [89–91] (*e.g., "in conclusion", "in summary", etc.*) that are often followed by the significant information. Therefore, a sentence including cue-phrase may be considered important. The score of cue-phrase feature is the ratio of the number of cue-phrases in the sentence to the sentence length (*number of tokens in the sentence*).

• Sentence location

The sentence location ranking is based on the idea that an important sentence can be judged by its position in the document or in a paragraph [92]. The authors usually introduce the main idea at the beginning of the document or the paragraphs [93]. Paragraphs at the beginning and at the end of a document are more informative and important to the document set [94]. To compute the sentence location score, we use the following equations [93]:

$$sentence\ position\ score\ relative\ to\ document\ (S_i) = 1-\frac{i-1}{N} \quad (15)$$

$$sentence\ position\ score\ relative\ to\ paragraph\ (S_i) = 1-\frac{i-1}{M} \quad (16)$$

where N and M are total numbers of sentences in document and paragraph respectively. $1<i<N,M$, where i is the ith sentence in the document or paragraph.

• Keyword weight

Key-word weight is an important feature that can be considered for text summarization. A sentence with a frequent word contains relevant information and can be indicative of the document topic [92,95]. The keywords are determined by Term Frequency (TF) and Inverse Document Frequency (IDF), as shown in Eq. (17) [9]. The TF is the number of times a word W_i appears in a document and the IDF is a measure of the general importance of the word. The IDF is calculated by dividing the total number of documents N by the number of documents, df_i, including the W_i and then taking the log of that quotient.

$$tf\ idf = TF_i\times IDF = TF_i\times\log\frac{N}{df_i} \quad (17)$$

The keyword weight score is the rate of summation of the TF–IDF of all words in a sentence over the maximum of summation values of all sentences in a document. It is calculated using the following equation:

$$Keyword\ weight\ (S_i) = \frac{\sum_{w_j\in S_i}(TFIDF)_{w_j}}{MAX\sum(TFIDF)} \quad (18)$$

• Proper Nouns and Numerals

These features are used to give importance to sentences. The sentence that includes more proper nouns and numerals is essential and likely to be included in the summary. For each sentence, we calculate the nouns feature score and numeric feature score as follows:

$$Nouns\ feature\ score = \frac{\#\text{ of proper noun in the sentence}}{\text{the length of the sentence}} \quad (19)$$

$$\text{Numerals } feature\ score = \frac{\#\text{ of numerals in the sentence}}{\text{the length of the sentence}} \quad (20)$$

• Sentence Length

The sentence length feature is employed to eliminate sentence that is too short in a summary since the short sentence is considered as not important for a summary. The sentence-length feature score is computed as follows:

$$sentence-length\ feature\ score = \frac{\#\ of\ tokens\ in\ the\ sentence, S}{\#\ of\ tokens\ in\ the\ long\ sentence} \quad (21)$$

• Non-essential information

Non-essential information indicates the sentence that includes some words such as "*because*", "*furthermore*", and "*additionally*". These words typically occur at the beginning of the sentence. If a sentence includes one of the non-essential words, its score equals to ((# *non-essential information words*)/ (*length of sentence*)); otherwise, 0.

• Sentence-to-Sentence similarity

This feature determines the importance of a sentence, *S*, based on the similarity measure between *S* and other sentences in a document (*i.e., X*). It is calculated as follows:

$$Sentence\ to\ sentence\ similarity\ (S_i) = \sum_{X \in D} \frac{Sim\ (S_i, X)}{\sum_{k \in D} Sim\ (k, X)} \quad (22)$$

where *D* includes all sentences. The $Sim\ (S_i, X)$ is calculated using the Jaccard measure (Jaccard, 1912), as shown in Eq. (23).

$$Sim\ (S_1, S_2) = \frac{\sum_{j=1}^{m}(w_{1j} \cdot w_{2j})}{\sum_{j=1}^{m} w_{1j}^2 + \sum_{j=1}^{m} w_{2j}^2 - \sum_{j=1}^{m}(w_{1j} \cdot w_{2j})} \quad (23)$$

where $S_1 = (w_{11}, w_{12,\ldots}, w_{1m})$ and $S_2 = (w_{21}, w_{22,\ldots}, w_{2m})$ are the vectors of sentences S_1 and S_2, respectively; w_{pj} is the weight of the *j*th word in vector S_p, *m* is the number of words. Each w_{pj} can be weighted using TF, TF–IDF, 0/1).

• Part-of-speech ratio

The numbers of nouns, verbs, adjectives and adverbs in the sentence, divided by the sentence length.

At the end of this step, we have a sentence-feature matrix. This feature matrix is learned using RBM algorithm. Finally, the final extracted feature vectors from the RBM will be fed into a SoftMax function to classify each sentence as opinion summary sentence or otherwise.

3.4. Opinion summarizer model (OSM)

The OSM contains two main phases: *sentence classification* and *Sentence selection*. The sentence classification phase concatenates the feature vector, *X*, extracted from the hidden layer of each embedding space (TSS and SAS) to form the final generated feature vector, $X = X_{TSS} \oplus X_{SAS} = [x_1, x_2, \ldots, x_m]$. The final vector representation, *X*, is then fed into the SoftMax classifier to predict label ("*opinion summary sentence*"; "*otherwise*") of each input sentence. The SoftMax function computes the probability distribution over labels using Eq. (24), where *X* is a sample vector, *K* is the number of labels (classes), *w* is the weight vector and *j* indicates the *j*th label.

$$P\ (y = j|X) = \frac{e^X}{\sum_{k=1}^{k} e^X}$$
$$X = x_0 w_0 + x_1 w_0 + \cdots + x_m w_m = \sum_{l}^{m} x_l w_l \quad (24)$$

In the *Sentence selection* phase, based on the sentence classification phase results, $S = \{s_1, s_2, \ldots., s_n\}$, where *n* is the number of sentences with assigned "*opinion summary sentence*" labels, the similarity score between each sentence $s_i \epsilon S$ and other sentences in *S* is calculated using Eq. (25).

$$Similarity\ score\ (s_i) = \sum_{d \in S} \frac{Sim\ (s_i, d)}{\sum_{l \in S} Sim\ (l, d)} \quad (25)$$

where, the similarity score between two sentences, $Sim\ (S_1, S_2)$ is computed using the Jaccard measure, Eq. (23). Let the $S_{sim\_score} = \{(s_1, val), (s_2, val) \cdots (s_n, val)\}$ includes all sentences and their corresponding similarity score, *val*.

In the next step, "*redundancy removal*", the greedy algorithm is employed to select a subset of sentences and eliminate the redundant information. To perform it, given the S_{sim_score}, first, the sentences are sorted in descending order. Then, a sentence with the highest similarity score is added into a summary. The next sentence is added into the summary, if its Jaccard similarity measure with other sentences of the summary does not exceed the pre-defined threshold. This process is repeated until a user-specified summary limit is reached. Finally, the summary is considered as an opinion summary.

4. Experimental results and discussion

As discussed in the literature, opinion summarization depends on sentiment analysis and text summarization aspects. For this reason, we incorporate both sentiment analysis and text summarization in our RDLS method. We create different models for each of them, namely: SAS and TSS. The idea is to train each model on its corresponding benchmark dataset and, then, use these pre-trained models together to extract opinion summary-related features from the datasets. The important research question here is – *does the RDLS model help to improve opinion summary generation performance?*". Throughout the rest of this section, we address the above-mentioned question in detail. In this section, we conducted three sets of experiments to verify the validity of the proposed method. Experiments 1, 2 and 3 are organized accordingly to evaluate the performance of SAS, TSS and RDLS.

4.1. The dataset

To have an extensive exploration of the RDLS, we conducted different experiments on various publicly available datasets that are specifically developed for sentiment analysis and text summarization. The Movie Review (MR) Dataset[4] [96] is a collection of movie-review documents labeled with respect to their overall sentiment polarity (positive or negative). The MR dataset includes 50,000 binary labeled reviews from IMDB (*positive 25,000, negative 25,000*). Besides than MR dataset, we also used Document Understanding Conferences (DUC) datasets, DUC[5] 2001 and 2002, which are commonly used evaluation corpora for summarization. The documents are all from the news domain and are grouped into various thematic clusters. The DUC 2002 dataset includes 567 documents on 59 different topics. The DUC 2001 dataset contains 60 sets of approximately 10 documents. It includes two main tasks: *(a)* single-document summarization: given a single document, a generic summary of the document with a length of approximately 100 words is created; *(b)* multi-document summarization: given a set of documents, four generic summaries of the documents with lengths of approximately 400, 200, 100, and 50 words are created.

To examine the performance of our method, we also need a gold standard data, which is a set of all correct results. We use the DUC 2002 dataset to create a gold standard dataset as

[4] http://www.cs.cornell.edu/people/pabo/movie-review-data/.

[5] http://duc.nist.gov.

judgment data to determine whether the output of the method is correct or not. For this purpose, three different annotators provided an extractive summary using the opinionated sentences as follows: *(1)* the text is decomposed into several sentences; *(2)* subsequently, the annotators specify the polarity of each sentence (*"Positive", "Negative", "Neutral"*); *(3)* the sentences are categorized into subjective and objective sentences; *(4)* an opinion summary is produced using the subjective sentences. In our experiment study, all the above data sets are randomly split into training set (70%) and test set (30%).

4.2. Performance measurement

We evaluate the RDLS using 2 Key Performance Indicators (KPI) which are: *(1)* Recall-Oriented Understudy for Gisting Evaluation (ROUGE-N) metric (Eq. (26)) (Lin, 2004) to judge the quality of a summary. ROUGE has been adopted by DUC as a standard automatic evaluation metric since 2004. The n-gram ROUGE metric can be computed as follows:

$$\text{ROUGE} - \text{N} = \frac{\sum_{S \epsilon \text{Reference summaries}} \sum_{\text{N-gram} \varepsilon S} \text{Count}_{\text{match}}(\text{N} - \text{gram})}{\sum_{S \epsilon \text{Reference summaries}} \sum_{\text{N-gram} \varepsilon S} \text{Count}(\text{N} - \text{gram})} \quad (26)$$

where N indicates the length of the n-gram and Count $_{\text{match}}$ (N-gram) indicates the total number of the n-grams occurring in both a reference and a candidate summary. Count (N-gram) indicates the number of n-grams in the reference summaries. In our experiments, we employed two metrics ROUGE-1 and ROUGE-2. We also measure the average ROUGE score value which is computed as (27).

$$\textit{Average ROUGE Score}\,(\textit{ARS}) = \frac{\text{ROUGE} - 1 + \text{ROUGE} - 2}{2} \quad (27)$$

(2) Accuracy where the proportion of correct answers over the total of answers considered in the evaluation.

4.3. Experiment 1: Sentiment analysis embedding space

The SAS model uses RNN–LSTM structure for sentiment feature extraction. We use the movie sentiment analysis dataset for the training. This pre-trained model is then employed in RDLS system. Each sentence of movie dataset is represented using the word embedding, sentiment and linguistic knowledge features. Then, RNN–LSTM receives the current feature vector and creates an output. Finally, in the output layer of RNN–LSTM, the concatenation of feature vector extracted from the last LSTM cell and the sentence-level feature is fed into a SoftMax fully connected layer. We use SoftMax function to classify data into two classes. This network is trained by stochastic gradient descent (*the optimization algorithm*) with cross-entropy (*negative log-likelihood*) loss functions. We use the learning rate of 0.03 to minimize the loss function. The cross-entropy is one of the common methods for evaluating the loss function [97]. The loss function[6] (*LF*) is computed using the cross-entropy formula as follows: $Lf\left(y, \tilde{y}\right) = -\sum y \ln(\tilde{y})$, where y is a gold distribution and $\tilde{y}$ is a predicted distribution (the model output distribution). In addition, in our experiment, we employ l^2 norm constraint regularization and dropout technique [98] to build a robust system. Dropout is an approach to regularization in neural networks which is employed to overcome overfitting problem. Dropout refers to ignoring neurons during the training phase of certain set of neurons which is chosen at random. The idea behind the dropout method is that during the training phase randomly sets hidden unit values to zero with a probability of p. The proportion of units to be dropped is a hyper-parameter to be determined by the user. We also impose a l^2 norm constraint during training for regularization [99]. L^2 norm[7] is calculated as the square root of the sum of the square vector values: $\|v\|_2 = sqrt\left(a_1^2, a_2^2, \ldots, a_n^2\right)$, where $\vec{v} = (a_1, a_2, \ldots, a_n)$.

Table 2
The results obtained from the experiments carried out varying the SAS method.

Methods	WEF	WLF	SLF	Accuracy
SAS $_{Full}$	+	+	+	86.13
SAS $_{WEF+SLF}$	+		+	82.21
SAS $_{WEF+WLF}$	+	+		84.50
SAS $_{WEF}$	+			79.67

4.3.1. Model variations: Effect of word-embedding, word and sentence levels features

To analyze the effect of the word embedding feature (WEF), sentence-level features (SLF), Word-level features (WLF), we train different SAS methods with different combinations of features as follows:

- SAS $_{\text{Full}}$: This method incorporates all the available features: WEF, SLF and WLF.
- SAS $_{\text{WEF+SLF}}$: This represents the SAS method in the absence of WLF feature set. It is trained and evaluated with the WEF and SLF features.
- SAS $_{\text{WEF+WLF}}$: A SAS method with WEF and WLF features.
- SAS $_{\text{WEF}}$: A SAS method that is trained and evaluated using WEF features.

The results of varying the SAS method are presented in Table 2. As shown in the table, among all the various SAS methods, the SAS $_{Full}$ achieves the best performance in comparison with the other methods with 86.13% accuracy. It outperforms SAS $_{WEF}$ by 6.46 points, SAS $_{WEF+WLF}$ by 1.63 points and reports approximately 4 points better accuracy as compared to SAS $_{WEF+SLF}$. Based on the result: *(a)* we get the best accuracy when we use all features. This can be explained by the fact that SAS $_{\text{Full}}$ exploits contextual information, sentiment information of sentences and words. It is also able to distinguish words with opposite sentiment polarity. *(b)* Word-level features have performed significantly better than sentence-level features. Due to the results, we use the combined features (*WEF + WLF + SLF*) for the proposed method.

4.3.2. Comparison with related methods

In this subsection, we compare our method, SAS, with the other existing well-known methods for sentence-level sentiment classification which are: *(1)* CNNSC [100] ; *(2)* IWVSA [31]; *(3)* DLUSA [101]; *(4)* CSNSA [102]; *(5)* TSND [2]; and *(6)* CCRSA [103]. Table 3 shows the performance of the SAS against these methods on the movie dataset in terms of accuracy. In Table 3, the column (*"SAS improvement"*) indicates the relative improvement for comparison between the SAS method and other methods. The relative improvement measure is computed as follows: $\left(\frac{\text{Our Method} - \text{Other method}}{\text{Other method}}\right) \times 100$. The sign "+" demonstrates the SAS improves the corresponding method. For instance, the SAS improves the performance of the TSND with 6.73% accuracy. From Table 3, it is found that DLUSA has the worst performance (*Accuracy: 76.20%*), while the SAS method (*Accuracy: 86.13%*) is extremely strong and performs best among all the methods. This stems from the following reasons:

[6] https://deeplearningdemystified.com/article/fdl-3.

[7] https://machinelearningmastery.com/vector-norms-machine-learning/.

Table 3
The accuracy comparisons between our method (SAS) and other methods, on MOV dataset.

Group	Methods	Accuracy	SAS improvement (%)
CNN	$CNNSC_{(CNN)}$	79.80	+ 7.93
	$IWVSA_{(CNN)}$	81.50	+ 5.68
DBN	$DLUSA_{(DBN)}$	76.20	+ 13.03
Recurrent	$SAS_{(LSTM)}$	86.13	–
	$CSNSA_{(bi\text{-}LSTM)}$	83.90	+ 2.66
Recursive	$TSND_{(Tree\ bi\text{-}LSTM)}$	80.70	+ 6.73
CNN-RNN	$CCRSA_{(CNN\text{-}LSTM)}$	82.20	+ 4.78

(1) Since *(a)* word2vec method does not explicitly exploit the sentiment information of the text, *(b)* a vector representation based on the word2vec method has not enough sentiment information to perform sentiment analysis and, *(c)* in vector representation using word2vec method, the words with opposite polarity such as "*good*" and "*bad*" are mapped to close word vectors, the SAS method incorporates the prior sentiment knowledge information (*a sentiment lexicon*) into the word2vec method to cope with the above-mentioned problems, while other methods (e.g., *TSND, CCRSA, CNNSC, DLUSA*) do not use the sentiment knowledge information.

(2) The SAS method also combines ten (10) sentiment dictionaries in order to use as the prior sentiment knowledge. The underlying reasons are to overcome the word coverage limit and various sentiment dictionaries complement each other. TSND, CCRSA, CNNSC, and DLUSA methods do not use any sentiment lexicon, CSNSA uses 4 sentiment dictionaries while IWVSA uses 6 sentiment dictionaries for sentiment analysis.

(3) In addition, since the word embedding model cannot differentiate the senses of a word and creates a single representation per word form, the SAS method can effectively deal with word sense variations, while the other methods (e.g., *CSNSA, TSND, CCRSA, CNNSC, DLUSA*) do not consider the aforementioned issue.

(4) Furthermore, since the performance of a sentiment analysis method relies on the different types of sentences, the SAS method considers the types of sentences in sentiment analysis, while the other methods do not consider them.

(5) The SAS also considers contextual polarity and the sentiment shifter rules (*refer to Section 3.1. Pre-processing*) in sentiment analysis, while the other methods (e.g., *CSNSA, TSND, CCRSA: excluding the sentiment shifter rules*) and (e.g., *DLUSA, CNNSC, IWVSA*: *excluding the sentiment shifter rules and contextual polarity*) do not consider them.

(6) RNN can handle sequential data while CNN cannot. Therefore, the SAS method can capture word order information and semantic relationships between words which are very important for text classification, while the other methods (e.g., *IWVSA, CNNSC, DLUSA*) cannot do so.

(7) Since word2vec method is good at capturing the semantic information, the addition of handcrafted features assists it in finding the sentiment more accurately (Table 3). The SAS method also uses complex external features (*word level and sentence levels features*), while the other methods do not use external features.

4.3.3. Sentiment lexicon size

We also investigate whether bigger sentiment lexicon can lead to promising results. The relationship between the number of lexicon words and the performance is presented in Fig. 7. We found that the smallest dictionaries (AFINN (*word size: 2477*)) perform poorly. There seems to be a correlation that larger is better. It can be observed that the SentiWordNet (*word size: 155,287*) leads to the highest throughput. The reason is due to the wide coverage of sentiment words.

Table 4
The results obtained from the experiments carried out varying the TSS model.

Methods	ROUGE-1	ROUGE-2	ARS
TSS_{Full}	0.3857	0.0864	0.2361
TSS_{SLK}	0.3682	0.0741	0.2212

Table 5
The results obtained from the experiments using TSS_{RBM} and TSS_{NORBM}.

Methods	ROUGE-1	ROUGE-2	ARS
TSS_{RBM}	0.3857	0.0864	0.2361
$TSS_{NO\ RBM}$	0.2875	0.0583	0.1729

4.4. Experiment 2: Text summarization embedding spaces

Extracting significant information from a text is a well-known challenging problem in text summarization. In our model, TSS, we use RBM algorithm to enhance and improve a set of features in order to extract informative information. The TSS model is trained using the DUC dataset to classify a sentence into binary classes, whether the sentence is a summary sentence or not. TSS embedding is trained as follows: first, the word embedding, statistical and linguistic knowledge feature vector is extracted for the text. Then, the feature set is passed through RBM. The feature vector extracted from RBM is used as input of a SoftMax fully connected layer. The nonlinearities of the hidden layers are corrected by a Rectified Linear Unit (ReLU) function. The output layer is a SoftMax that contains two neurons to classify a sentence. Training of network is performed by minimizing the binary cross-entropy error. For parameter optimization, we use stochastic gradient descent procedure with the learning rate set to 0.03. For regularization, we apply a dropout mechanism and a constraint on l^2 norms of the weight vectors to the network.

4.4.1. Model variations: Effect of word-embedding, statistical and linguistic knowledge features

In this subsection, we compare TSS_{Full} with TSS_{SLK}, where *SLK* (*indicates statistical and linguistic knowledge feature*) and *Full* (*refers to SLK + WEF: statistical and linguistic knowledge features combined with word embedding-based feature*). In this experiment, our aim is to examine the efficiency of the combination of *SLK* and *WEM* on TSS model. To do this, we conduct an experiment on DUC 2001 dataset. The experimental results of $TSS_{WEF+SLK}$ and TSS_{SLK} are presented in Table 4. From the results, we find the $TSS_{WEF+SLK}$ greatly outperforms TSS_{SLK} (ARS: 0.2361). It shows that *(1)* WEF vector can help the SLK vector to learn better sentence representations; *(2)* the combination of them improves the overall performance; *(3)* the outstanding summarization ability of the $TSS_{WEF+SLK}$. However, due to the experiment, we use the combined features (*SLK + WEF*) for the TSS model.

4.4.2. Model analysis: Effect of restricted Boltzmann machine (RBM)

In the current subsection, we investigate the effect of RBM on TSS. To do this, we compare $TSS_{NO\ RBM}$ with TSS_{RBM} as follows:

(1) TSS_{RBM} considers RBM and the input feature vector is given as input to RBM.

(2) TSS_{NORBM} does not consider RBM and the input feature vector is given as input to fully connected layer (refer to Fig. 5).

The experimental results of these methods are presented in Table 5. It is found that the encouraging performance on this dataset is obtained when the TSS model uses RBM algorithm. In other words, we find that TSS_{RBM} (*ARS: 0.2361*) performs better than the TSS_{NORBM} (*ARS: 0.1729*) model to produce a text summary.

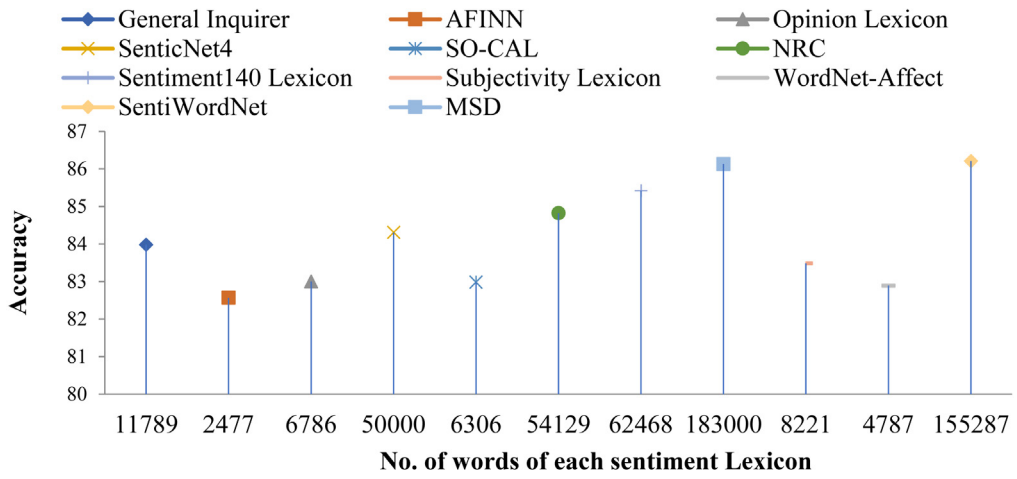

Fig. 7. Performance and the size of sentiment dictionary words.

Table 6
Performance comparison of extractive methods on the DUC 2001 dataset.

Methods	ROUGE-1	ROUGE-2	ARS	TSS improvement (%)
LexRank	0.3343	0.0609	0.1976	+ 19.46
Reg Manual	0.3455	0.0718	0.2087	+ 13.13
TSS_{Full}	0.3857	0.0864	0.2361	–
RASR	0.3631	0.0849	0.2240	+ 5.38
R2N2(GA)	0.3588	0.0764	0.2176	+ 8.48
PriorSum	0.3598	0.0789	0.2194	+ 7.61
R2N2(ILP)	0.3691	0.0787	0.2239	+ 5.43

4.4.3. Comparison with related methods

To evaluate our model, an extensive experiment on TSS_{Full} was carried out. We compare TSS_{Full} with the systems that performed best on DUC 2001 dataset, which are: *(1)* graph-based summarization model LexRank [104]; *(2)* PriorSum which uses a CNN and surface features to encode each sentence [105]; *(3)* Reg Manual [105] which employs human-compiled features such as '*number*', '*nentity*' and '*stopratio*'; *(4)* RASR [106], a system to model importance and redundancy simultaneously by computing the relative importance of a sentence given a set of selected sentences; *(5)* R2N2(GA) and R2N2(ILP) [107] where the R2N2 uses a recursive neural network to learn the combination of hand-crafted features.

Results of TSS model against other methods are listed in Table 6, mainly measured by ARS. In Table 6, each row represents a summarization method and each column explains an evaluation metric. From the results presented in the table, the model TSS_{Full} obtains the highest performance in terms of *ARS* metric on the dataset (ARS: 0.2361). In other words, the TSS significantly outperforms state-of-the-art summarization methods R2N2, PriorSum, etc. The improvement of our method comes from two aspects: *(1)* the effectiveness of the incorporation of word embedding, statistical and linguistic knowledge feature. This means that the vector representation of each sentence using the above-mentioned features has more input information which can be critical for determining how important the sentence is for the document. In other words, to determine whether a sentence should be included in the final summary or not; *(2)* the TSS_{Full} deals with redundancy and information diversity issues.

4.5. Experiment 3: Opinion summarizer model

In this section, we evaluate the performance of OSM. As discussed before, the OSM includes two main phases: *(1)* a sentence classification phase (SCP) based on the DNN to predict the label of each input sentence; and *(2)* a sentence selection phase (SSP) to create final opinion summary. In the SCP, the concatenation of feature vectors extracted from the SAS and TSS models is used as input of a SoftMax fully connected layer. The output layer is a SoftMax that contains two neurons to classify a sentence (*"opinion summary sentence"*; *"otherwise"*). Training of the DNN is performed by minimizing the binary cross-entropy error. The back-propagation algorithm with the learning rate of 0.03 is used to update weights. Furthermore, we also apply the dropout technique and a constraint on l^2 norms of the weight vectors to the network for regularization.

4.5.1. Model Analysis: Effect of 'Sentiment analysis embedding space', 'Text summarization embedding spaces' and 'Sentiment analysis embedding space + Text summarization embedding spaces'

In the current subsection, we investigate the effects of TSS (OSM_{TSS}), SAS (OSM_{SAS}) and the 'SAS + TSS' (OSM_{Full}) to produce an opinion summary. We eliminate the TSS model and SAS model from OSM_{Full} separately and conduct three different experiments on DUC 2002 dataset:

***(1)* TSS (OSM_{TSS}):** we apply the TSS model and obtain the vector representation from the last hidden layer. Subsequently, we pass the current vector to the SCP of OSM to predict label of each input sentence. Finally, the output of the SCP is passed to the SSP in order to produce the final summary.

***(2)* SAS (OSM_{SAS}):** we apply the SAS model and obtain the vector representation from the last hidden layer. Subsequently, we perform the same above-mentioned processes.

***(3)* 'SAS + TSS' (OSM_{Full}):** we apply the combination of the SAS and TSS models to the dataset. The experimental results of these approaches are presented in Table 7.

As can be seen from the results in Table 7, the SAS model (ARS: 0.2003) is more effective than TSS model. This is because by employing SAS model, the OSM_{SAS} obtains relevant sentences which include expressions of opinion. In addition, the vector representation extracted from SAS model can extract the deep sentiment information, which contributes more to sentence classification using SCP.

We also see the contribution of TSS (ARS: 0.1294) by removing the SAS model. It is interesting to find that TSS is obviously lower effective than SAS. This can be described by the fact that in contrast to OSM_{SAS}, when the OSM uses only the TSS model, many sentences that participated in the final summary contain no opinions.

Table 7
The results obtained from the experiments carried out varying the OSM.

Methods	TSS	SAS	ROUGE-1	ROUGE-2	ARS
OSM_{TSS}	+		0.2273	0.0315	0.1294
OSM_{SAS}		+	0.3269	0.0737	0.2003
OSM_{Full}	+	+	0.4673	0.0859	0.2766

Table 8
The Performance of the OSM_{Full} against other methods.

Methods	ROUGE-1	ROUGE-2	ARS	OSM improvement (%)
OMSHR	0.3831	0.0571	0.2201	+ 25.66
SOSML	0.4161	0.0679	0.2420	+ 14.30
TSAD	0.3334	0.0451	0.1892	+ 46.16
OSM_{Full}	0.4673	0.0859	0.2766	–
CHOS	0.3832	0.0461	0.2146	+ 28.86
SOSPR	0.3041	0.0401	0.1721	+ 60.72

As a result, the encouraging performance on this dataset is obtained when the SAS model is used together with the TSS model (ARS: 0.2766). In other words, we find that the jointed architecture of TSS and SAS models performs better than the TSS and SAS models alone to produce the final summary.

4.5.2. Comparison with related methods

In order to demonstrate the robustness of the OSM_{Full}, we compare it with several existing well-known methods on the aforementioned dataset. These methods are: *(1)* OSUG [58]; *(2)* SUSA [57]; *(3)* OSHR [108]; *(4)* OSPR [109]; and *(5)* SOSML [28]. The experimental comparison between OSM_{Full} and other methods is presented in Table 8. We observe from table that OSM_{Full} achieves ARS of 0.2766 in this dataset, which is a better performance than some other methods. We believe that there are several reasons for the OSM_{Full} to outperform other methods. We discuss the reasons for the effectiveness of OSM_{Full} as follows.

(1) OSM_{Full} takes into account contextual polarity, types of a sentence and subjectivity analysis in sentiment analysis. *(2)* Unlike other methods, OSM_{Full} integrates multi sentiment lexicons to tackle the maximum coverage problem of words. Furthermore, it also exploits the SSM approach to specify word sentiment score if it does not exist in MSD. *(3)* It integrates several valuable resource-information such as sentiment, statistical and linguistic knowledge. It also considers redundancy removal and information diversity issues to improve the quality of a summary. *(4)* OSM_{Full} encodes the sentiment information into the word embedding features since word embedding algorithm only model the syntactic context of words and ignores the sentiment of text. *(5)* The augmentation of other features (e.g., *sentiment knowledge-based, statistical and linguistic knowledge-based features*) using word2vec that can extract the deep semantic relationships between words. *(6)* Unlike other methods, OSM_{Full} can effectively deal with word sense variations. *(7)* It takes advantage of both the RBM model and the RNN–LSTM model, thus achieve higher performance than the existing methods. Both of them, which are non-linear in nature, better fit the data than other methods. On the other hand, since word order information and semantic relationships between words have an important impact on sentiment classification performance, the RNN–LSTM is able to handle time-series data and learn the long-term dependencies.

5. Conclusion and future work

Recently, the deep learning-based method has attracted lots of attention in many fields such as image processing, speech recognition and natural language processing. In this paper, we present a novel deep-learning-based method for the generic opinion-oriented extractive summarization of multi-documents, RDLS. The RDLS includes three main parts: *(1)* SAS: to extract relevant features in order to determine sentence polarity; *(2)* TSS: to extract relevant features in order to identify important sentence; and *(3)* OSM: the concatenation features extracted from TSS and SAS are then passed through OSM to produce an opinion summary. The RDLS incorporates various types of features which are sentiment, statistical, linguistic knowledge and word-level embedding to create a hybrid vector representation of each sentence. It also combines several sentiment dictionaries to tackle the word coverage limit. In addition, the RDLS considers several strategies to address the following problems: *(a)* contextual polarity; *(b)* sentiment shifter; *(c)* type of sentence; *(d)* word sense variations; *(e)* integration of sentiment information and word embedding; *(f)* word order information and semantic relationships between words.

We conduct several experiments on three open benchmark datasets, DUC 2002, DUC 2001 and Movie datasets. The experimental results on these datasets confirm the effectiveness of the RDLS. The results also show that the method is competitive with state-of-the-art to previously reported results. We also conduct some comparisons between:

(1) (OSM_{TSS}), SAS (OSM_{SAS}) and ($OSM_{SAS+TSS}$): based on the results, the combination of the *SAS* and *TSS* could provide good performance in terms of ARS measure.

(2) $TSS_{SLK+WEF}$ and TSS_{SLK}: the experimental results show the $TSS_{WEF+SLK}$ greatly outperforms TSS_{SLK}.

(3) different SAS methods including $SAS_{WLF+SLF+WEF}$, $SAS_{WEF+SLF}$, $SAS_{WEF+WLF}$, SAS_{WEF}: the results show among all the various SAS methods, the $SAS_{Full:\ WLF+SLF+WEF}$ achieves the best performance in comparison with the other methods.

In future work, we aim to study the influence of different similarity measures (*Sentence-to-Sentences similarity*) on performance of the method. In addition, we also aim to consider the passive and active voice in the comparison between two sentences, since the current method is not able to distinguish between an active sentence and a passive sentence. Given three sentences (A: '*Teacher likes his student.*'; B: '*student likes his teacher.*'; C: '*student is liked by his teacher.*'), although the similarity measure between sentences (A and B) and (A and C) are the same, as we can see, the meaning of sentence A is more similar to the sentence C. Hence, it is important to know what passive and active sentences are before comparisons can be conducted. We are also interested to investigate in more depth the efficiency of comparative and sarcastic sentences on RDLS and measure the performance of the RDLS using large and various datasets. Finally, we would like to consider a rich set of new features in order to get more relevant sentences and meaningful summary sentences.

CRediT authorship contribution statement

Asad Abdi: Conceptualization, Methodology, Investigation, Software, Writing - original draft. **Shafaatunnur Hasan:** Data curation, Writing - review & editing. **Siti Mariyam Shamsuddin:** Supervision, Funding acquisition. **Norisma Idris:** Data curation, Writing - review & editing. **Jalil Piran:** Software, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by The Ministry of Higher Education (MOHE), Malaysia under "Q.J130000.21A2.03E53 - statistical machine learning method to text summarization", "13H82- intelligent predictive analytics for the retail industry", "04G48- HIR 4.1: scalable analytics algorithm for digital intelligence environment", "03G91- QR big data code on the cloud" and "17H62- GPUMLIB: big data streaming for visualization analytics". The authors would like to thank Research Management Centre (RMC), ASEAN-India Collaborative R&D Program (AISTDF), Cyber–Physical System research group, as well as School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia (UTM) for their continuous support in making this research a success.

References

[1] E. Cambria, Affective computing and sentiment analysis, IEEE Intell. Syst. 31 (2016) 102–107.
[2] J. Li, M.-T. Luong, D. Jurafsky, E. Hovy, When are tree structures necessary for deep learning of representations? in: arXiv e-prints, 2015, pp. 2304–2314.
[3] C. Song, X.-K. Wang, P.-f. Cheng, J.-q. Wang, L. Li, SACPC: A Framework based on probabilistic linguistic terms for short text sentiment analysis, Knowl.-Based Syst. (2020) 105572.
[4] M.A. Hassonah, R. Al-Sayyed, A. Rodan, A.-Z. Ala'M, I. Aljarah, H. Faris, An efficient hybrid filter and evolutionary wrapper approach for sentiment analysis of various topics on Twitter, Knowl.-Based Syst. 192 (2020) 105353.
[5] A. Abdi, S.M. Shamsuddin, S. Hasan, J. Piran, Deep learning-based sentiment classification of evaluative text based on multi-feature fusion, Inf. Process. Manage. 56 (2019) 1245–1259.
[6] B. Mutlu, E.A. Sezer, M.A. Akcayol, Multi-document extractive text summarization: A comparative assessment on features, Knowl.-Based Syst. 183 (2019) 104848.
[7] M.A. Mosa, A.S. Anwar, A. Hamouda, A survey of multiple types of text summarization with their satellite contents based on swarm intelligence optimization algorithms, Knowl.-Based Syst. 163 (2019) 518–532.
[8] N. Chatterjee, P.K. Sahoo, Random indexing and modified random indexing based approach for extractive text summarization, Comput. Speech Lang. 29 (2015) 32–44.
[9] R.M. Alguliyev, R.M. Aliguliyev, N.R. Isazade, An unsupervised approach to generating generic summaries of documents, Appl. Soft Comput. 34 (2015) 236–250.
[10] E. Lloret, E. Boldrini, T. Vodolazova, P. Martínez-Barco, R. Muñoz, M. Palomar, A novel concept-level approach for ultra-concise opinion summarization, Expert Syst. Appl. 42 (2015) 7148–7156.
[11] Q. Li, Z. Jin, C. Wang, D.D. Zeng, Mining opinion summarizations using convolutional neural networks in Chinese microblogging systems, Knowl.-Based Syst. 107 (2016) 289–300.
[12] A. Abdi, S.M. Shamsuddin, S. Hasan, J. Piran, Automatic sentiment-oriented summarization of multi-documents using soft computing, Soft Comput. 23 (2019) 10551–10568.
[13] P. Gupta, R. Tiwari, N. Robert, Sentiment analysis and text summarization of online reviews: A survey, in: Communication and Signal Processing (ICCSP), 2016 International Conference on, IEEE, 2016, pp. 0241–0245.
[14] X. Du, Y. Cai, S. Wang, L. Zhang, Overview of deep learning, in: Chinese Association of Automation (YAC), Youth Academic Annual Conference of, IEEE, 2016, pp. 159–164.
[15] S. Sun, C. Luo, J. Chen, A review of natural language processing techniques for opinion mining systems, Inf. Fusion 36 (2017) 10–25.
[16] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.
[17] M. Giatsoglou, M.G. Vozalis, K. Diamantaras, A. Vakali, G. Sarigiannidis, K.C. Chatzisavvas, Sentiment analysis leveraging emotions and word embeddings, Expert Syst. Appl. 69 (2017) 214–224.
[18] P.J. Stone, E.B. Hunt, A computer approach to content analysis: studies using the general inquirer system, in: Proceedings of the May 21–23, 1963, Spring Joint Computer Conference, ACM, 1963, pp. 241–256.
[19] F. Årup Nielsen, A new ANEW: Evaluation of a word list for sentiment analysis in microblogs, in: arXiv e-prints, 2011.
[20] T. Chen, R. Xu, Y. He, X. Wang, Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN, Expert Syst. Appl. 72 (2017) 221–230.
[21] N. Majumder, S. Poria, H. Peng, N. Chhaya, E. Cambria, A. Gelbukh, Sentiment and sarcasm classification with multitask learning, IEEE Intell. Syst. 34 (2019) 38–43.
[22] X. Sun, C. Li, F. Ren, Sentiment analysis for Chinese microblog based on deep neural networks with convolutional extension features, Neurocomputing 210 (2016) 227–236.
[23] Y. Cai, Q. Huang, Z. Lin, J. Xu, Z. Chen, Q. Li, Recurrent neural network with pooling operation and attention mechanism for sentiment analysis: A multi-task learning approach, Knowl.-Based Syst. (2020) 105856.
[24] N. Liu, B. Shen, Aspect-based sentiment analysis with gated alternate neural network, Knowl.-Based Syst. 188 (2020) 105010.
[25] A. Abdi, S.M. Shamsuddin, R.M. Aliguliyev, QMOS: Query-based multi-documents opinion-oriented summarization, Inf. Process. Manage. 54 (2018) 318–338.
[26] B. Liu, Sentiment analysis and opinion mining, Synth. Lect. Hum. Lang. Technol. 5 (2012) 1–167.
[27] V.B. Raut, D. Londhe, Opinion mining and summarization of hotel reviews, in: 2014 International Conference on Computational Intelligence and Communication Networks, IEEE, 2014, pp. 556–559.
[28] A. Abdi, S.M. Shamsuddin, S. Hasan, J. Piran, Machine learning-based multi-documents sentiment-oriented summarization using linguistic treatment, Expert Syst. Appl. 109 (2018) 66–85.
[29] B. Pang, L. Lee, 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, 2004, p. 271.
[30] A. Mudinas, D. Zhang, M. Levene, Combining lexicon and learning based approaches for concept-level sentiment analysis, in: Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining, ACM, 2012, p. 5.
[31] S. Mahdi Rezaeinia, A. Ghodsi, R. Rahmani, Improving the accuracy of pre-trained word embeddings for sentiment analysis, in: arXiv e-prints, 2017.
[32] M.-Y. Day, Y.-D. Lin, Deep learning for sentiment analysis on google play consumer review, in: Information Reuse and Integration (IRI), 2017 IEEE International Conference on, IEEE, 2017, pp. 382–388.
[33] A. Caliskan, J.J. Bryson, A. Narayanan, Semantics derived automatically from language corpora contain human-like biases, Science 356 (2017) 183–186.
[34] Y. Liu, B. Liu, L. Shan, X. Wang, Modelling context with neural networks for recommending idioms in essay writing, Neurocomputing 275 (2018) 2287–2293.
[35] N. Jindal, B. Liu, Identifying comparative sentences in text documents, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2006, pp. 244–251.
[36] E. Riloff, J. Wiebe, Learning extraction patterns for subjective expressions, in: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, 2003, pp. 105–112.
[37] S. Babar, P.D. Patil, Improving performance of text summarization, Procedia Comput. Sci. 46 (2015) 354–363.
[38] R.M. Alguliyev, R.M. Aliguliyev, N.R. Isazade, A. Abdi, N. Idris, COSUM: Text summarization based on clustering and optimization, Expert Syst. 36 (2019) e12340.
[39] S. Verma, V. Nidhi, Extractive summarization using deep learning, in: arXiv e-prints, 2017.
[40] Y. Zhang, M.J. Er, R. Zhao, M. Pratama, Multiview convolutional neural networks for multidocument extractive summarization, IEEE Trans. Cybern. 47 (2017) 3230–3242.
[41] M. Yousefi-Azar, L. Hamey, Text summarization using unsupervised deep learning, Expert Syst. Appl. 68 (2017) 93–105.
[42] J.-P. Qiang, P. Chen, W. Ding, F. Xie, X. Wu, Multi-document summarization using closed patterns, Knowl.-Based Syst. 99 (2016) 28–38.
[43] S.P. Singh, A. Kumar, A. Mangal, S. Singhal, Bilingual automatic text summarization using unsupervised deep learning, in: Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on, IEEE, 2016, pp. 1195–1200.
[44] J. Carbonell, J. Goldstein, The use of MMR, diversity-based reranking for reordering documents and producing summaries, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 1998, pp. 335–336.
[45] A. Abdi, N. Idris, R. Alguliyev, R. Aliguliyev, Query-based multi-documents summarization using linguistic knowledge and content word expansion, Soft Comput. (2015) 1–17.
[46] X. Wan, J. Xiao, Graph-based multi-modality learning for topic-focused multi-document summarization, in: IJCAI, 2009, pp. 1586–1591.
[47] Y. Ouyang, W. Li, S. Li, Q. Lu, Applying regression models to query-focused multi-document summarization, Inf. Process. Manage. 47 (2011) 227–237.

[48] S.-h. Zhong, Y. Liu, B. Li, J. Long, Query-oriented unsupervised multi-document summarization via deep learning model, Expert Syst. Appl. 42 (2015) 8146–8155.
[49] M. Denil, A. Demiraj, N. de Freitas, Extraction of salient sentences from labelled documents, in: arXiv e-prints, 2014.
[50] K. Duraiswamy, An approach for text summarization using deep learning algorithm, 1 (2014) 1–9.
[51] N. Kurian, S. Asokan, Summarizing user opinions: A method for labeled-data scarce product domains, Procedia Comput. Sci. 46 (2015) 93–100.
[52] R.E.L. Condori, T.A.S. Pardo, Opinion summarization methods: Comparing and extending extractive and abstractive approaches, Expert Syst. Appl. 78 (2017) 124–134.
[53] M. Gambhir, V. Gupta, Recent automatic text summarization techniques: a survey, Artif. Intell. Rev. 47 (2017) 1–66.
[54] A. Sadh, A. Sahu, D. Srivastava, R. Sanyal, S. Sanyal, Extraction of relevant figures and tables for multi-document summarization, Comput. Linguist. Intell. Text Process. (2012) 402–413.
[55] M. Mladenović, J. Mitrović, C. Krstev, D. Vitas, Hybrid sentiment analysis framework for a morphologically rich language, J. Intell. Inf. Syst. 46 (2016) 599–620.
[56] H. Saggionα, A. Funk, Interpreting SentiWordNet for opinion classification, in: Proceedings of the Seventh Conference on International Language Resources and Evaluation LREC10, 2010.
[57] N. Yadav, N. Chatterjee, Text summarization using sentiment analysis for DUC data, in: Information Technology (ICIT), 2016 International Conference on, IEEE, 2016, pp. 229–234.
[58] A. Balahur, M. Kabadjov, J. Steinberger, R. Steinberger, A. Montoyo, Challenges and solutions in the opinion summarization of user-generated content, J. Intell. Inf. Syst. 39 (2012) 375–398.
[59] T.K. Landauer, On the computational basis of learning and cognition: Arguments from LSA, Psychol. Learn. Motiv. 41 (2002) 43–84.
[60] S. Kim, R. Calvo, Sentiment-oriented summarisation of peer reviews, in: Artificial Intelligence in Education, Springer, 2011, pp. 491–493.
[61] M. Kabadjov, A. Balahur, E. Boldrini, Sentiment intensity: Is it a good summary indicator? in: Language and Technology Conference, Springer, 2009, pp. 203–212.
[62] R. Xia, F. Xu, J. Yu, Y. Qi, E. Cambria, Polarity shift detection elimination and ensemble: A three-stage model for document-level sentiment analysis, Inf. Process. Manage. 52 (2016) 36–45.
[63] Y. Xie, Z. Chen, K. Zhang, Y. Cheng, D. Honbo, A. Agrawal, A.N. Choudhary, MuSES: MUltilingual sentiment elicitation system for social media data, IEEE Intell. Syst. 29 (2014) 34–42.
[64] H. Nguyen, M.-L. Nguyen, A deep neural architecture for sentence-level sentiment classification in Twitter social networking, in: International Conference of the Pacific Association for Computational Linguistics, Springer, 2017, pp. 15–27.
[65] Z. Xiao, X. Li, L. Wang, Q. Yang, J. Du, A.K. Sangaiah, Using convolution control block for Chinese sentiment analysis, J. Parallel Distrib. Comput. 116 (2018) 18–26.
[66] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in: European Conference on Machine Learning, Springer, 1998, pp. 137–142.
[67] M. Bansal, K. Gimpel, K. Livescu, Tailoring continuous word representations for dependency parsing, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2014, pp. 809–815.
[68] C.-C. Lin, W. Ammar, C. Dyer, L. Levin, Unsupervised POS induction with word embeddings, in: arXiv e-prints, 2015, pp. 1311–1316.
[69] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: arXiv e-prints, 2013..
[70] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.
[71] A.M. Schaefer, S. Udluft, H.-G. Zimmermann, Learning long-term dependencies with recurrent neural networks, Neurocomputing 71 (2008) 2481–2488.
[72] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (1994) 157–166.
[73] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[74] M. Kamkarhaghighi, M. Makrehchi, Content tree word embedding for document representation, Expert Syst. Appl. 90 (2017) 241–249.
[75] S.M. Mohammad, S. Kiritchenko, X. Zhu, NRC-Canada: building the state-of-the-art in sentiment analysis of tweets, in: arXiv e-prints, 2013, pp. 321–327.
[76] E. Cambria, S. Poria, D. Hazarika, K. Kwok, SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[77] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, M. Stede, Lexicon-based methods for sentiment analysis, Comput. Linguist. 37 (2011) 267–307.
[78] S. Baccianella, A. Esuli, F. Sebastiani, SentiWordNet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining, in: LREC, 2010, pp. 2200-2204.
[79] F.H. Khan, U. Qamar, S. Bashir, A semi-supervised approach to sentiment analysis using revised sentiment strength based on SentiWordNet, Knowl. Inf. Syst. (2016) 1–22.
[80] M. Hu, B. Liu, Mining and summarizing customer reviews, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2004, pp. 168–177.
[81] C. Strapparava, A. Valitutti, WordNet Affect: an affective extension of WordNet, in: LREC, Citeseer, 2004, pp. 1083–1086.
[82] P. Jaccard, The distribution of the flora in the alpine zone, New Phytol. 11 (1912) 37–50.
[83] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507.
[84] P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory, in: Colorado Univ at Boulder Dept of Computer Science, 1986, pp. 194–281.
[85] T. Yamashita, M. Tanaka, E. Yoshida, Y. Yamauchi, H. Fujiyoshii, To be Bernoulli or to be Gaussian, for a restricted Boltzmann machine, in: 2014 22nd International Conference on Pattern Recognition, IEEE, 2014, pp. 1520–1525.
[86] K. Cho, T. Raiko, A. Ilin, Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines, in: ICML, 2011.
[87] J. Zhang, H. Wang, J. Chu, S. Huang, T. Li, Q. Zhao, Improved Gaussian–Bernoulli restricted Boltzmann machine for learning discriminative representations, Knowl.-Based Syst. 185 (2019) 104911.
[88] J. Kupiec, J. Pedersen, F. Chen, A trainable document summarizer, in: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 1995, pp. 68–73.
[89] L. Alonso, Representing discourse for automatic text summarization via shallow NLP techniques (Tesis doctoral), Universitat de Barcelona, Barcelona, 2005.
[90] B. Fraser, What are discourse markers? J. Pragmat. 31 (1999) 931–952.
[91] A. Knott, A data-driven methodology for motivating a set of coherence relations, 1996.
[92] J.L. Neto, A.A. Freitas, C.A. Kaestner, Automatic text summarization using a machine learning approach, in: Brazilian Symposium on Artificial Intelligence, Springer, 2002, pp. 205–215.
[93] C.S. Yadav, A. Sharan, Hybrid approach for single text document summarization using statistical and sentiment features, Int. J. Inf. Retr. Res. 5 (2015) 46–70.
[94] S. Teufel, M. Moens, Sentence extraction as a classification task, in: Proceedings of the ACL, 1997, pp. 58–65.
[95] A. Nenkova, L. Vanderwende, K. McKeown, A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2006, pp. 573–580.
[96] A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Association for Computational Linguistics, 2011, pp. 142–150.
[97] C. Bishop, C.M. Bishop, Neural Networks for Pattern Recognition, Oxford university press, 1995.
[98] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, in: arXiv e-prints, 2012.
[99] Y. Zhang, B. Wallace, A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, 2015, arXiv e-prints, arXiv:1510.03820.
[100] Y. Kim, Convolutional neural networks for sentence classification, 2014, arXiv e-prints, arXiv:1408.5882.
[101] M. Di Capua, A. Petrosino, A deep learning approach to deal with data uncertainty in sentiment analysis, in: International Workshop on Fuzzy Logic and Applications, Springer, 2016, pp. 172–184.
[102] Z. Teng, D.T. Vo, Y. Zhang, Context-sensitive lexicon features for neural sentiment analysis, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1629–1638.
[103] X. Wang, W. Jiang, Z. Luo, Combination of convolutional and recurrent neural network for sentiment analysis of short texts, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 2428–2437.

[104] G. Erkan, D.R. Radev, Lexrank: Graph-based lexical centrality as salience in text summarization, J. Artif. Intell. Res. 22 (2004) 457–479.
[105] Z. Cao, F. Wei, S. Li, W. Li, M. Zhou, W. Houfeng, Learning summary prior representation for extractive summarization, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2015, pp. 829–833.
[106] P. Ren, F. Wei, C. Zhumin, M. Jun, M. Zhou, A redundancy-aware sentence regression framework for extractive summarization, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 33–43.
[107] Z. Cao, F. Wei, L. Dong, S. Li, M. Zhou, Ranking with recursive neural networks and its application to multi-document summarization, in: AAAI, 2015, pp. 2153-2159.
[108] V.B. Raut, D. Londhe, Opinion mining and summarization of hotel reviews, in: Computational Intelligence and Communication Networks (CICN), 2014 International Conference on, IEEE, 2014, pp. 556–559.
[109] S. Mac Kim, R.A. Calvo, Sentiment-oriented summarisation of peer reviews, in: International Conference on Artificial Intelligence in Education, Springer, 2011, pp. 491–493.