Software Impacts 18 (2023) 100582

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Semantic-Summarizer: Semantics-based text summarizer for English language text

Mudasir Mohd [a], Nowsheena [a], Mohsin Altaf Wani [a], Hilal Ahmad Khanday [a,*], Umar Bashir Mir [b], Sheikh Nasrullah [c], Zahid Maqbool [d], Abid Hussain Wani [a]

[a] South Campus University of Kashmir, India
[b] Lal Bahadur Shastri Institute of Management, India
[c] IBM Research Almaden, United States of America
[d] Higher Education J&K, India

ARTICLE INFO

Keywords:
Text summarization
Extractive text summarization
Semantic models
Word2Vec

ABSTRACT

Text summarization is a process that condenses text documents for efficient information consumption. It offers numerous benefits, including time savings, reduced information overload, informed decision-making, and even content generation. In this paper, we present *SemanticSum*, an advanced text document summarizer that not only achieves compression but also preserves the underlying semantics of the original text. By leveraging semantic analysis, our summarizer intelligently identifies and removes redundant sentences that express the same meaning in different forms. Our experimental evaluation demonstrates that *SemanticSum* outperforms several state-of-the-art open-source summarizers regarding summary quality. The results validate the effectiveness of our approach, which capitalizes on semantics to produce more accurate and contextually meaningful summaries.

Code metadata

Current code version	*V1*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2023-374
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/8179578/tree/v1
Legal Code License	*GNU General Public License v3.0*
Code versioning system used	*git*
Software code languages, tools, and services used	*Python and Flask*
Compilation requirements, operating environments & dependencies	*Python*3.6 >=
If available Link to developer documentation/manual	https://github.com/mudasirmohd/sematic-Text-Summarizer.git
Support email for questions	mudasir.mohammad@kashmiruniversity.ac.in

1. Introduction

Unstructured textual data management is a critical problem in the domain of Natural language processing (NLP). With the ever-increasing volume of digital textual data, the attention needed to consume this data is difficult to manage, and thus, automatic text summarization (ATS) is needed [1]. ATS reduces the volume of textual documents by compressing them while preserving their overall information content. ATS aims to reduce the length to retain the maximum information content. This tradeoff is difficult to maintain and requires a deeper analysis of textual documents. The semantic analysis provides deeper, concrete insights into the underlying meaning of the text documents. Our *SemanticSum* uses the semantic information contained in the sentences of a document to produce semantically rich text summaries. We

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: mudasir.mohammad@kashmiruniversity.ac.in (M. Mohd), nowsheenab@gmail.com (Nowsheena), mohsinwani@uok.edu.in (M.A. Wani), hilalkhanday@kashmiruniversity.ac.in (H.A. Khanday), mirumar.iitd@gmail.com (U.B. Mir), Nasrullah.sheikh@ibm.com (S. Nasrullah), zahidcomp@gmail.com (Z. Maqbool), abid.wani@uok.edu.in (A.H. Wani).

https://doi.org/10.1016/j.simpa.2023.100582
Received 21 August 2023; Received in revised form 10 September 2023; Accepted 17 September 2023

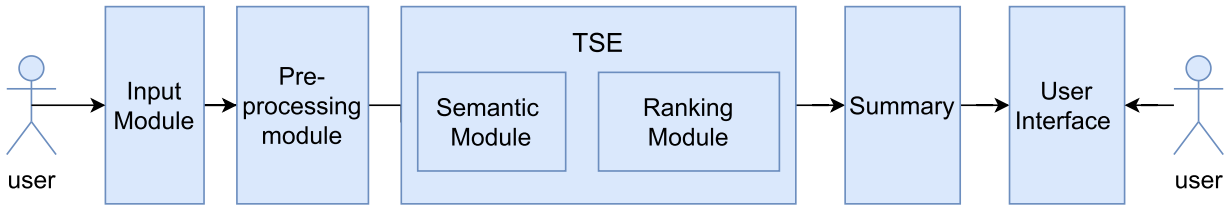

Fig. 1. Semantic extension of tweets.

use semantics and other stylistic and syntactic features to produce more meaningful and coherent summaries.

Text summarization is important and essential in information processing and extraction. It enables users to grasp key points quickly and the given text's main ideas. It finds its applications in various domains. It enables users to read effectively and efficiently. Summarized information improves the attention time of the reader. It has been observed that lengthy texts result in a reduced attention span of the reader, and thus, summarized content can lead to increased attention cycles [2]. Summarization leads to improved question-answering systems [3]. Summaries lead to reduced reading time and, hence, increased efficacy of readers. Efficient summaries help during research as it reduces and simplifies the search time of researchers. Summaries are also used for efficient customer reviewing systems. Thus, we find an excellent motivation for designing and improving text summarizers for producing good summaries.

SemanticSum is an English language text summarizing system. It uses distributional semantic algorithms to obtain summaries. We use these algorithms to compute the semantic coherence of two textual elements and then use them as a feature for summarization.

SemanticSum is built on the premise that semantics is an essential and inherent text structure. If semantics is used along with the other features of text, the overall summarization process will improve. Semantic feature usage is lacking in the previous state-of-art conducted in the area of ATS. Moreover, the results confirm our hypothesis, and we conclude that using semantic features improves the overall process of ATS.

2. Functionalities

SemanticSum is written in Python. It uses different machine learning libraries like scikit-learn, pytorch and gensim. The front end of *SemanticSum* is written in Flask.

SemanticSum has two components: the backend of the system, which we call The Summarizing Engine (TSE), and the frontend is for user interaction. Users input their documents through the front end, which TSE summarizes. TSE is the core of the software. It processes and analyzes the text and then generates summaries. The generated summaries are then displayed to the user through the system's web interface. The system also takes as input multiple files stored in the folder and then generates summaries of each individual file. We describe the functionalities and the structure of both the components of *SemanticSum* in the following subsections.

2.1. The Summarizing Engine (TSE)

TSE is our software's core and main part that summarizes the given text documents. It uses state-of-the-art Natural language (NLP) processing tools and machine learning algorithms to obtain the summary of the input text document. The documents written in English are the input for TSE; we then perform semantic analysis and extensions of each sentence in the document. These semantic extensions are then clustered into semantic spaces, and finally, we use our novel ranking algorithm to generate the summaries. Fig. 1 shows the overall structure of our TSE graphically.

Precisely, TSE summarizes the input text document by the following modules: (1) Preprocessing module: This module receives text from the input source, removes the inconsistencies, and converts text in a uniform format. (2) The semantic module uses distributional semantic algorithms to convert text into semantic extensions. We use Word2Vec [4] for capturing semantics.Word2Vec is a distributional semantic model, and its usage in capturing text semantics is beneficial since it is used in several studies in the literature. Canales et al. [5] used Word2Vec for the emotion analysis task. They used it to create an automatically annotated corpus to determine emotion polarity. Mohd et al. [6] it for extending the reach of lexicons in the sentiment polarity detection task. Schütze [7] used Word2Vec in word-sense disambiguation problem. Rafiya Jan et al. [8] used it to improve the performance of their emotion classifier. Our semantic module uses our novel *big-vector* generation algorithm for creating semantic extensions of each sentence in a text document. (3) We next cluster these semantic extensions of each sentence to a semantic space using clustering algorithms. Our clustering module performs this clustering. (4) Finally, we apply our novel ranking algorithm to assign ranks to each sentence in each cluster. The ranking module uses these scores to obtain n-sentences from different clusters to obtain the summary. Each of these modules is explained next.

2.1.1. Preprocessing module

The preprocessing module is responsible for removing inconsistencies in the data and converting data into a uniform format. Preprocessing helps in the efficient processing of data. It uses the following steps:

- **Removing Stopwords:** To extract the syntactic features, the text is scanned for stopwords presence. Particularly, articles and prepositions are removed from the text. We remove stopwords to reduce noise and remove less significant words from the text.
- **Removing punctuations:** Punctuations are also noise for summarization tasks, and thus, we remove all the punctuations from the text before processing it.
- **Lowercase:** For text uniformity, we convert all the text in the input document into lowercase.
- **Lemmatization:** Lemmatization is the process through which the words of the text are converted to their base forms. These base forms are such that they retain their meaning. We perform lemmatization to make the text uniform. This uniformity helps in the proper processing of text.
- **Removing URL's:** URL present in the text are removed before processing.

2.1.2. Semantic module

The semantics of text describe the text structure and its underlying meaning. While summarizing the documents, semantics are not preserved [9]. Thus, existing summarizers in the literature [10] overlook this aspect of the text. *SemanticSum* preserves text semantics and uses it as a feature for text summarization. We use semantic features in combination with other stylistic and syntactic features to obtain summaries. We employ distributional semantic algorithms to extract semantic features from the text. In particular semantic module uses Word2Vec [4]. These models are independent of linguistic analysis

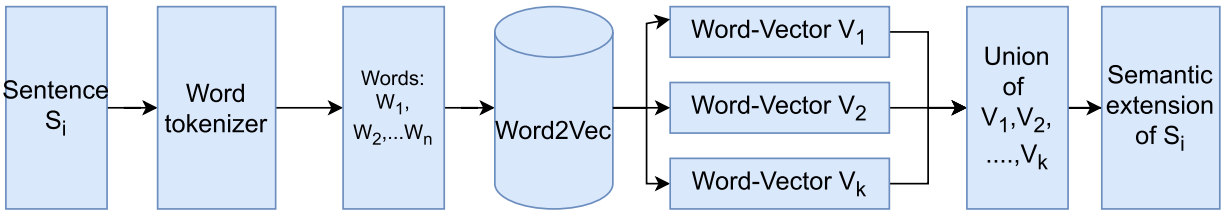

Fig. 2. Semantic extension of sentences.

and thus serve as an excellent means to extract semantic information. Furthermore, these models are trained to be domain-independent.

SemanticSum uses our novel big-vector generation algorithm that utilizes Word2Vec to obtain the semantics of text. Big-vectors are used to generate semantic extensions of text. It provides mapping to transform textual data into its semantically rich form. These big-vectors are used in different studies like email summarization [11], election prediction [12], sentiment analysis [6]. Fig. 2 shows the process of generation of big-vectors diagrammatically.

All the sentences 'S' of the input document 'D' are fed to the Word2Vec to obtain their big-vectors. Let $\delta(w)$ be a function for retrieving a top list of 'm' words from a semantic model. The function is given as $w' = \delta(w) = w'_1 \oplus w'_2 \oplus, \ldots, \oplus w'_m$. For a sentence with $W = \{w_1, w_2, w_3, \ldots., w_k\}$ as the sequence of k tokenized words, a big-vector BGV is populated by concatenating respective top m similar words for each word i.e $BGV = \{\delta(w_1) \oplus \delta(w_2) \oplus, \ldots, \oplus \delta(w_k)\}$.

The algorithm for Big-vector generation is given in the algorithm 1.

Algorithm 1 Algorithm for Big-vector generation

Let D be the input document
s_i is the sentences of document D

for all $s_i \in$ D **do**
  $W \leftarrow Tokenization(s_i)$
  where $W = \{w_1, w_2, w_3, ..., w_n\}$

  for all $w_i \in W$ **do**
    $V_i \leftarrow Word2Vec(W_i)$
  end for
  $BV = V_1 \oplus V_2 \oplus \cdots \oplus V_{|W|}$
  $\oplus$ is concatenation
end for

2.1.3. Clustering module

The clustering module of *SemanticSum* receives big-vectors generated by the semantic module. Clustering is the process of grouping data into similar groups. Data is grouped together based on some similar attributes that it shares.

Upon receiving the big-vectors of all sentences, the clustering module groups these big-vectors into semantically similar clusters. We use K-Means [13] clustering algorithm for clustering. These clusters are then fed to the ranking module to extract top-n sentences that eventually form the summary of the document.

2.1.4. Ranking module

The purpose of the ranking module is to rank each sentence of every semantic cluster. The semantic module generates the semantic clusters (Section 2.1.3). Our ranking module then uses stylistic and other syntactic features of text to rank each sentence from different clusters. The ranking module's features include TF-IDF, proper nouns, cosine-similarity, and sentence position.

Ranking module calculates different scores according to these features for each sentence. These individual scores are then summed to obtain the total normalized score for each sentence. The ranks are then calculated based on this score, and thus, we obtain a summary by extracting top-n sentences as per their ranks.

2.2. Web interface

The system incorporates a user-friendly web interface that enables seamless interaction with the system. This interface is developed using the Flask framework, providing a robust foundation for web application building. The web interface design follows a minimalist approach, ensuring a clean and intuitive user experience.

Within the web interface, users are able to input the desired text that they want to summarize. Once the user submits the text, the Text Summarization Engine (TSE) works, leveraging its powerful algorithms and techniques to generate a concise summary of the input. This summary, the end result of the process, is then presented back to the user through the web interface, allowing them to access and utilize the generated summary conveniently.

3. Impacts

SemanticSum is our software that produces summaries of English language text documents. It uses semantics as a feature for generating text summaries. Summaries generated by the software are semantically enriched and are of high quality. The competitive superiority of our software is proved by the comparative analysis with the four baseline state-of-the-art open-source summarizers. For obtaining comparative analysis, we used the DUC-2007 dataset, available on demand[1]. The summaries generated using our approach, as well as the baselines, are assessed against the ground truth using various ROUGE metrics, including Precision (Pr), Recall (Re), and F-Score (Fs). We generated both short and long summaries for extensive evaluation. The short summary length is 15%, and the long summary length is 25%. The average of all ROUGE metrics is calculated for all documents in the dataset and is presented in the Tables 1 and 2 for summary lengths 15% and 25% respectively.

The results of the summarization experiment clearly demonstrate that our proposed system outperforms the state-of-the-art baselines in precision and F-Score. Figs. 3 and 4 show the summarization process results.

SemanticSum plays a vital role in combating information overload by generating concise and semantically meaningful text summaries. By leveraging semantics and preserving the original structure of the text, this method effectively alleviates the burden of information overload, enabling users to comprehend the core message and save precious time rapidly. Swiftly grasping a text's essence through a summary empowers users to prioritize their actions more efficiently.

SemanticSum has been employed in various studies and projects. In the paper by [9], our algorithm was used to generate summaries for the DUC-2007 dataset, resulting in semantically enhanced summaries that outperformed several open-source alternatives available online. This achievement was significant because these results paved the way for more coherent and semantically correct text summarization systems. In another study [6], our software addressed the issue of zero-padding, ensuring consistent-length data samples for LSTM-based sentiment classifiers. This consistent input length significantly improved the accuracy and reduced bias in the sentiment classification results.

[1] https://duc.nist.gov/duc2007/tasks.html

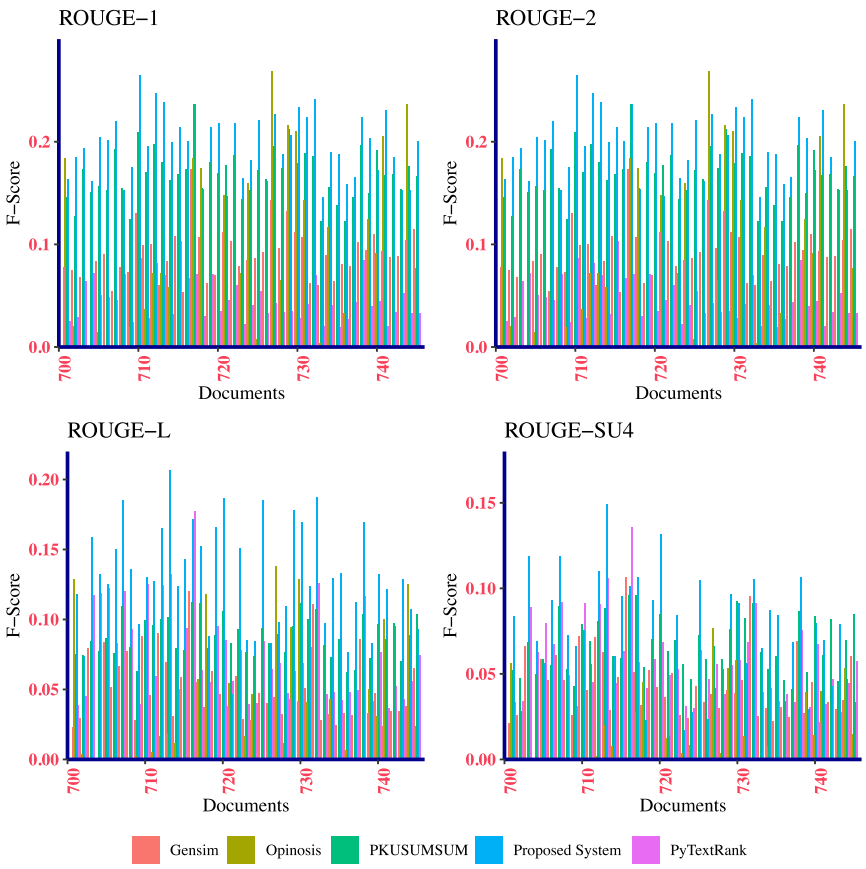

Fig. 3. F-Score for 15% summary length.

Table 1
Averaged summarization results of 15% summary length.

Metric	Rouge type	SemanticSum	Gensim	OPINOSIS	PKUSUMSUM	PyTeaser
Pr	ROUGE-1	**0.34** (0.04)	0.05 (0.02)	0.19 (0.17)	0.10 (0.007)	0.030 (0.016)
	ROUGE-2	**0.07** (0.02)	0.02 (0.01)	0.03 (0.05)	0.04 (0.08)	0.097 (0.058)
	ROUGE-L	0.20 (0.03)	0.05 (0.01)	0.05 (0.07)	0.10 (0.01)	**0.44** (0.024)
	ROUGE-SU4	**0.13** (0.02)	0.03 (0.014)	0.09 (0.096)	0.05 (0.007)	0.033 (0.016)
Rc	ROUGE-1	0.34 (0.10)	**0.84** (0.14)	0.07 (0.02)	0.74 (0.05)	0.120 (0.024)
	ROUGE-2	0.08 (0.04)	**0.44** (0.11)	0.01 (0.01)	0.28 (0.06)	0.701 (0.080)
	ROUGE-L	0.20 (0.04)	**0.47** (0.18)	0.05 (0.07)	0.49 (0.04)	0.369 (0.075)
	ROUGE-SU4	0.14 (0.15)	**0.52** (0.11)	0.02 (0.04)	0.39 (0.05)	0.275 (0.0081)
Fs	ROUGE-1	**0.33** (0.06)	0.09 (0.05)	0.08 (0.11)	0.17 (0.01)	0.046 (0.020)
	ROUGE-2	0.07 (0.03)	0.01 (0.01)	0.01 (0.02)	**0.17** (0.02)	0.163 (0.079)
	ROUGE-L	**0.20** (0.03)	0.09 (0.02)	0.07 (0.08)	0.17 (0.02)	0.075 (0.033)
	ROUGE-SU4	**0.13** (0.03)	0.05 (0.01)	0.03 (0.04)	0.09 (0.01)	0.056 (0.023)

The real-world significance of *SemanticSum* is further evident in its adoption by industries, as demonstrated by its integration into the software application available at[2]. During the final industrial phase, users can conveniently access email summaries directly on their mobile devices through a dedicated mobile app. This seamless integration not only boosts user productivity but also highlights the practicality and effectiveness of *SemanticSum* in professional settings.

4. Conclusion and future improvements

SemanticSum can be used by the users to summarize the text and reduce information overall. It produces summaries of text that the users consume in a faster time. The essence of *SemanticSum* is that it uses

[2] https://quickwordz.com

Table 2
Averaged summarization results of 25% summary length.

Metric	Rouge Type	SemanticSum	Gensim	OPINOSIS	PKUSUMSUM	PyTextRank
Pr	ROUGE-1	**0.112** (0.014)	0.048 (0.013)	0.064 (0.156)	0.075 (0.013)	0.097 (0.024)
	ROUGE-2	0.140 (0.051)	0.043 (0.027)	0.189 (0.178)	0.049 (0.008)	**0.317** (0.042)
	ROUGE-L	**0.110** (0.016)	0.024 (0.012)	0.088 (0.096)	0.028 (0.005)	0.109 (0.026)
	ROUGE-SU4	**0.104** (0.007)	0.021 (0.009)	0.028 (0.054)	0.024 (0.005)	0.071 (0.023)
Rc	ROUGE-1	0.248 (0.066)	0.521 (0.188)	0.053 (0.075)	**0.649** (0.061)	0.106 (0.021)
	ROUGE-2	0.791 (0.111)	**0.875** (0.066)	0.070 (0.117)	0.850 (0.044)	0.404 (0.052)
	ROUGE-L	0.451 (0.114)	**0.557** (0.088)	0.025 (0.043)	0.508 (0.066)	0.165 (0.035)
	ROUGE-SU4	0.354 (0.012)	**0.476** (0.103)	0.011 (0.019)	0.421 (0.079)	0.095 (0.081)
F1	ROUGE-1	**0.150** (0.015)	0.087 (0.023)	0.067 (0.081)	0.134 (0.022)	0.101 (0.022)
	ROUGE-2	0.230 (0.047)	0.080 (0.045)	0.080 (0.116)	0.092 (0.014)	**0.355** (0.045)
	ROUGE-L	**0.171** (0.010)	0.045 (0.021)	0.029 (0.043)	0.053 (0.010)	0.136 (0.029)
	ROUGE-SU4	**0.153** (0.019)	0.039 (0.171)	0.012 (0.020)	0.045 (0.010)	0.081 (0.027)

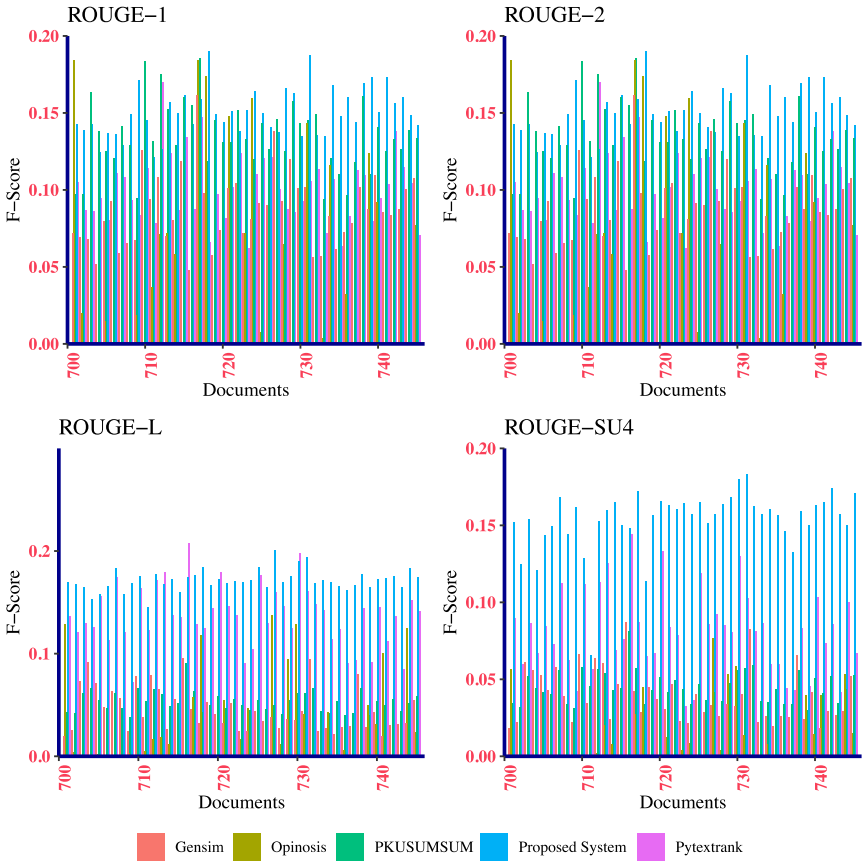

Fig. 4. F-Score for 25% summary length.

semantics as a feature for summarization, thereby preserving underlying text structure during summarization. The results produced by the proposed software achieve competitive superiority against the state-of-art baselines. Thus proving our hypothesis that semantic features aid in text summarization. In the forthcoming iterations of the software, our objective is to enhance *SemanticSum* by making it capable of processing multiple languages. Our ongoing efforts in creating semantic models for low-resource languages will pave the way for developing similar models for these languages. As a result, we will be able to extend the functionality of *SemanticSum* to encompass multilingual text processing.

Declaration of competing interest

None

References

[1] M. Mohd, M.B. Shah, S.A. Bhat, U.B. Kawa, H.A. Khanday, A.H. Wani, M.A. Wani, R. Hashmy, Sumdoc: A unified approach for automatic text summarization, in: Proceedings of Fifth International Conference on Soft Computing for Problem Solving: SocProS 2015, Vol. 1, Springer, 2016, pp. 333–343.

[2] I.K. Bhat, M. Mohd, R. Hashmy, Sumitup: A hybrid single-document text summarizer, in: Soft Computing: Theories and Applications: Proceedings of SoCTA 2016, Vol. 1, Springer, 2018, pp. 619–634.

[3] M. Mohd, R. Hashmy, Question classification using a knowledge-based semantic kernel, in: Soft Computing: Theories and Applications: Proceedings of SoCTA 2016, Vol. 1, Springer, 2018, pp. 599–606.

[4] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, arXiv preprint arXiv:1301.3781.

[5] L. Canales, C. Strapparava, E. Boldrini, P. Martínez-Barco, Intensional learning to efficiently build up automatically annotated emotion corpora, IEEE Trans. Affect. Comput. (2017).

[6] M. Mohd, S. Javeed, Nowsheena, M.A. Wani, H.A. Khanday, Sentiment analysis using lexico-semantic features, J. Inf. Sci. (2022) 01655515221124016.

[7] H. Schütze, Automatic word sense discrimination, Comput. Linguist. 24 (1) (1998) 97–123.

[8] R. Jan, A.A. Khan, Emotion mining using semantic similarity, Int. J. Synth. Emot. (IJSE) 9 (2) (2018) 1–22.

[9] M. Mohd, R. Jan, M. Shah, Text document summarization using word embedding, Expert Syst. Appl. 143 (2020) 112958.

[10] M. Kirmani, N. Manzoor Hakak, M. Mohd, M. Mohd, Hybrid text summarization: A survey, in: K. Ray, T.K. Sharma, S. Rawat, R.K. Saini, A. Bandyopadhyay (Eds.), Soft Computing: Theories and Applications, Springer Singapore, Singapore, 2019, pp. 63–73.

[11] M. Kirmani, G. Kaur, M. Mohd, ShortMail: An email summarizer system, Softw. Impacts 17 (2023) 100543, http://dx.doi.org/10.1016/j.simpa.2023.100543, URL https://www.sciencedirect.com/science/article/pii/S2665963823000805.

[12] M. Mohd, S. Javeed, M.A. Wani, H.A. Khanday, A.H. Wani, U.B. Mir, S. Nasrullah, Poliweet — Election prediction tool using tweets, Softw. Impacts 17 (2023) 100542, http://dx.doi.org/10.1016/j.simpa.2023.100542, URL https://www.sciencedirect.com/science/article/pii/S2665963823000799.

[13] J.A. Hartigan, M.A. Wong, Algorithm AS 136: A k-means clustering algorithm, J. R. Stat. Soc. Ser. C (Appl. Stat.) 28 (1) (1979) 100–108.