Unsupervised Extractive Text Summarization using Frequency-based Sentence Clustering

Ali Hajjar[1] and Joe Tekli[1[0000-0003-3441-7974]]

[1]School of Engineering, Lebanese American University (LAU)
36 Byblos, Lebanon
ali.hajjar@lau.edu, joe.tekli@lau.edu.lb

Abstract. Large texts are not always entirely meaningful: they might include repetitions and useless details, and might not be easy to interpret by humans. Automatic text summarization aims to simplify text by making it shorter and (possibly) more informative. This paper describes a new solution for extractive text summarization, designed to efficiently process flat (unstructured) text. It performs unsupervised frequency-based document processing to identify the candidate sentences having the highest potential to represent informative content in the document. It introduces a dedicated feature vector representation for sentences to evaluate the relative impact of different sentence terms. The sentence feature vectors are run through a partitional k-means clustering process, to build the extractive summary based on the cluster representatives. Experimental results highlight the quality and efficiency of our approach.

Keywords: Automatic text summarization, extractive summaries, word space model, feature representation, k-means clustering.

1. Introduction

The exponential increase of data published on the Web has reignited interest in automatic text summarization, aiming to save data storage space and allow faster access to the most informative data. In this study, we introduce a new solution for extractive summarization of flat (unstructured) text, by integrating and adapting different existing techniques in a novel way, aiming to provide a simple, flexible, and computationally efficient solution. While most extractive solutions utilize term-based feature vectors with heuristic or linear optimization solutions, our solution performs sentence feature vector extraction, followed by sentence clustering using their feature vector similarity, and summary building based on the cluster representatives. The user selects the input text document, and the size of the final summary expressed in number of sentences. The input text is processed for term frequency computation in order to produce a co-occurrence matrix describing the feature vector of each sentence in the original text. The sentence feature vectors are utilized to compute pair-wise sentence similarities, in order to perform partitional *k*-means clustering to group similar sentences together, where the number of *k* clusters corresponds to the size of the summary provided initially by the user. Different strategies are suggested to select the most representative sentences from the clusters, to form the output summary. Experiments highlight our solution's quality and almost linear computation time.

The remainder of the paper is organized as follows. Section 2 reviews the related works. Section 3 described our proposal. Section 4 describes our experimental evaluation, before concluding in Section 5 with ongoing works and future directions.

2. Related Works

Automatic text summarization techniques can be grouped in two main categories: abstractive and extractive. Abstractive summarization aims to generate summaries the way humans perform summarization, by transforming the original text to generate a new text summary. They usually follow a predefined schema describing a certain ordered pattern of content organization [1, 2]. The schema can be expressed using knowledge-based or rule-based representations. Knowledge-based approaches use of a machine-readable semantic graph made of a set of concepts representing word senses, and a set of links representing semantic relations (synonymy, hyponymy, etc., [3, 4]). A text document is represented as a semantic sub-graph, and the process of summarization consists in generating a reduced semantic sub-graph using some heuristic rules, in order to generate the reduced output summary. Rule-based approaches describe text relationships using typed dependencies between pairs of words. Rule-based information extraction and content selection heuristics [5-7] are then used to generate new texts, based on training data consisting of sets of input texts and expected (summarized) output texts. Text chunks from the original text are matched against the rules, and sent to a generation module trained based on the training data, in order to produce the output summary. Recent approaches have utilized deep learning transformer based encoder-decoder architectures like BERT to produced trained summarization models, e.g., [8-10]. Deep learning solutions have shown promising results compared with their counterparts [8], albeit requiring training data and training time which are not always available.

Extractive summarization promotes a less complex process of identifying and extracting the most informative text tokens, without content re-writing or generation. Most techniques in this context use unsupervised term frequency computation, e.g., [11, 12], identifying and combining the sentences or paragraphs including the most frequent terms to form the summary. This is based on the assumption that the high frequency of specific words in a text may be a good indicator of its significance. As a result, text chunks or sentences are compared based on their most common terms. Heuristic or linear optimization solutions can be used to identify the most representative text chunks or sentences, to be combined into the output summary. The authors in [13] use an adapted quantum-inspired genetic algorithm, using a modified quantum measurement and a self-adaptive quantum rotation gate based on the quality and length of the summary. The authors in [14] use a swarm optimization solution, using word mover distance and normalized Google distance to evaluate text similarity. The authors in [15] use a projected gradient descent algorithm to perform summarization through convex optimization. In [16], the authors represent sentences as nodes in an undirected graph, where every distinct term is represented as a vertex, regardless of the number of term occurrences. The process extracts the most connected nodes in the graph, to form the output summary. Some approaches, e.g., [2, 17], perform feature vector transformation using singular value decomposition (SVD) or latent derelict analysis (LDA), to represent terms or sentences in a latent semantic space. Sentence selection is then conducted in the latent semantic space, before compiling the sentences in their original form to construct the output summary.

While most extractive solutions utilize term-based feature vectors combined with heuristic or linear optimization solutions, we introduce a new sentence-based feature vector representation combined with a partitional clustering algorithm, aiming to improve summarization efficiency and quality.

3. Proposal

Our solution consists of four main components (cf. Fig. 1): i) linguistic pre-processing, ii) sentence feature vector representation, iii) sentence clustering, and iv) summary building.

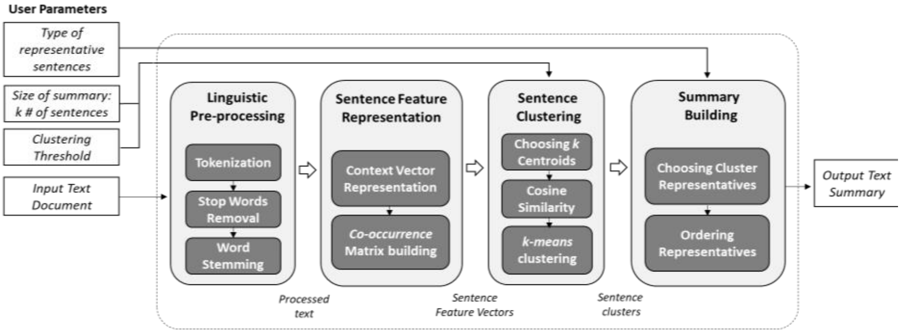


Fig. 1. Simplified activity diagram describing our approach

3.1. Linguistic Pre-Processing

A sequence of preprocessing tasks is first executed before the documents can be processed for sentence feature extraction, clustering, and summarization. First, this component converts all words to their lowercase form, performs tokenization to distinguish separate terms, and removes stop-words from the obtained term sequences. Second, it performs stemming or lemmatization, following the user's preference: i) stemming converts all the words to their original syntactic forms (stems) using syntactic stemming rules[1]; ii) lemmatization transforming words into their original lexical forms using a lexical reference[2]. Third, it performs sentence extraction based on text punctuations, where each sentence is represented as a sequence of stemmed/lemmatized terms.

3.2. Sentence Feature Vector Representation

Different from most existing approaches which rely on term frequencies in processing text documents (cf. Section 2), we introduce a sentence-based feature vector representation to capture the syntactic similarities between sentences. We adopt sentences as our base extractive summarization unit, and aim to identify the most informative sentences to put in the output summary. The pseudocode for our sentence feature representation component is shown in Fig. 2. It accepts as input: a text document to be summarized, and produces as output: the set of sentence feature vectors for all sentences in the document. For each sentence s_i in the document, the algorithm builds a square matrix M_i designed to store the co-occurrence scores of all terms in s_i (cf. Fig. 2, lines 1-5). Each line k in M_i represents the context vector of term t_k, denoted as the centroid of line k. Each column ℓ represents a term t_ℓ in the context of centroid t_k. The term co-occurrence weight of t_ℓ in the context of t_k is computed according to the relative distance of t_ℓ from t_k:

[1] We use the Porter Stemmer in our approach since it is one the most effective in the literature.
[2] We use the WordNet lexical dictionary [3] to perform lemmatization, due to its common usage in the literature.

$$w^k_\ell = \begin{cases} \text{if } (\ell < k) & \dfrac{\ell}{k-1} \\ \text{if } (\ell = k) & 0 \\ \text{else} & 1-\left(\dfrac{\ell-k-1}{m-k}\right) \end{cases} \quad \textbf{(1)}$$

where *m* is the number of terms in the sentence, *ℓ* and *k* represent the occurrence indices of terms t_ℓ and t_k in the sequence of term tokens representing sentence s_i (cf. linguistic preprocessing in Section 3.1). Once weights in the co-occurrence matrix are computed (lines 6-10), the algorithm aggregates the weights of identical terms occurring multiple times in the sentence (lines 11-15). It then creates a reduced feature vector including only the distinct terms as vector dimensions (lines 16-18), and aggregates the weights from all centroid context vectors (i.e., from all lines in the co-occurrence matrix) to from the output sentence feature vector (lines 19-21).

```
Algorithm Sentence_Feature_Vector_Representation
Input:  D       // Text document
Output: Vs      // Set of sentence feature vectors

Begin
S = {s1, s2, …, sm}   // set of sentences in D, each represented as a sequence of term tokens     1
Vs = ϕ                // set of sentence feature vectors in D                                     2
For each sentence si in S                                                                          3
    Ti = {t1, t2, …, tm}                                          // set of terms in si            4
    Initialize co-occurrence matrix Mi of dimensions m^2          // term co-occurrence matrix     5
    For each line k in M                                                                           6
        Consider term tk as sentence centroid                                                      7
        For each column ℓ in Mi                                   // weight of tℓ in context of tk 8
            Compute $w^k_\ell$ following formula (1)                                               9
        Compute $context(t_k) = [w^k_1, \ldots, w^k_k{=}0, \ldots, w^k_m]$   // context vector of centroid tk   10
    Initialize reduced co-occurrence matrix Mi* of dimension n^2  // n is number of distinct terms in S   11
    For each line ℓ in Mi*                                                                         12
        Compute aggregate weight $w_\ell = \sum_{\forall t_j \in \text{occurrence of } t_\ell} (w_j, M_i)$   13
    For each column k in Mi*                                                                       14
        Compute aggregate weight $w_k = \sum_{\forall t_j \in \text{occurrence of } t_k} (w_j, M_i)$         15
    Initialize sentence feature vector Vi of dimension n                                           16
    For each column k in Vi                                                                        17
        Compute aggregate weight $agg(t_k, M^*_i) = \sum_{\forall t_j \in context(t_k)} w^k_j$               18
    Normalize weights in Vi                                     // divide by sum of aggregate weights  19
    Add Vi to Vs                                                // other normalizations can be used    20
Return Vs                                                                                          21
End
```

Fig. 2. Pseudocode of the sentence feature vector representation component

Consider the following sentence, extracted from one of our test documents (cf. Section 5): *A solar eclipse occurs when the Moon passes between Earth and the Sun, thereby totally* *or partially obscuring Earth's view of the Sun*. The tokenized representation of the sentence (following linguistic pre-processing, cf. Section 3.1) is:

Term indices	1	2	3	4	5	6	7	8	9	10
	solar	**eclipse**	**occurrence**	**Moon**	**passage**	**Earth**	**Sun**	**obscurity**	**Earth**	**Sun**

where the number of terms *m*=10. Consider the first term *solar* as centroid, i.e., *k*=1:

Term indices	1	2	3	4	5	6	7	8	9	10
	solar	**eclipse**	**occurrence**	**Moon**	**passage**	**Earth**	**Sun**	**obscurity**	**Earth**	**Sun**
Context weights	0	9/9	8/9	7/9	6/9	5/9	4/9	3/9	2/9	1/9

centroid
ℓ=k

ℓ>k

The weight of context term *eclipse* having ℓ = 2 (> *k*) is $1-\frac{\ell-k-1}{m-k}=1-\frac{2-1-1}{10-1}=1$. It is the highest weight in this context vector since it represents the closest term to the centroid. The weights decrease gradually as the terms occur farther away form the centroid, reaching minimum weight $=\frac{1}{9}$ for the last term *Sun*.

Consider the forth term *Moon* as centroid, i.e., *k*=4:

Term indices	1	2	3	4	5	6	7	8	9	10
	solar	**eclipse**	**occurrence**	**Moon**	**passage**	**Earth**	**Sun**	**obscurity**	**Earth**	**Sun**
Context weights	1/3	2/3	3/3	0	6/6	5/6	4/6	3/6	2/6	1/6

ℓ<k

centroid
ℓ=k

ℓ>k

Here, the weight of term *eclipse* having ℓ = 2 (< *k*) is $\frac{\ell}{k-1}=\frac{2}{4-1}=\frac{2}{3}$. The weight of term *passage* having ℓ = 5 (> *k*) is $1-\frac{\ell-k-1}{m-k}=1-\frac{5-4-1}{10-4}=1$. The weights decrease gradually as the terms occur farther away from the centroid, reaching minimum weight = $\frac{1}{3}$ for the last term to the left, *solar*, and minimum weight = $\frac{1}{6}$ for the last term to the right, *Sun*. The complete co-occurrence matrix for all centroids is shown in Table 1. Table 2.a shows the reduced co-occurrence where term repetitions (highlighted in color in Table 1) have been aggregated. Table 2.b shows the output sentence feature vector, normalized w.r.t.[1] the sum of the aggregate scores (other normalization functions can be used, following user preferences). The latter highlights the weight of every term in the sentence, considering its relative potion w.r.t. all other terms as well as its number of repetitions in the sentence.

3.3. Sentence Clustering

Once the sentence feature vectors are computed, we utilize *k*-means partitional clustering to group similar sentences together, where the sentence clusters serve as seeds for extractive summarization. We adopt *k*-means as one of the most prominent and efficient algorithms, allowing the user to choose and control the size of the summary, where the number of

[1] With respect to

clusters *k* represent the number of sentences in the output summary. Yet, other clustering algorithms can be used (e.g., *k*-medians, *k*-medoids, and constrained partitional clustering where the number of output clusters is chosen by the user [18]).

Table 1. Sample square co-occurrence matrix

Centroid terms / *Context terms* / *Context vectors of all centroid terms*

	solar	eclipse	occurrence	Moon	passage	Earth	Sun	obscurity	Earth	Sun
solar	0	9/9	8/9	7/9	6/9	5/9	4/9	3/9	2/9	1/9
eclipse	1/1	0	8/8	7/8	6/8	5/8	4/8	3/8	2/8	1/8
occurrence	1/2	2/2	0	7/7	6/7	5/7	4/7	3/7	2/7	1/7
Moon	1/3	2/3	3/3	0	6/6	5/6	4/6	3/6	2/6	1/6
passage	1/4	2/4	3/4	4/4	0	5/5	4/5	3/5	2/5	1/5
Earth	1/5	2/5	3/5	4/5	5/5	0	4/4	3/4	2/4	1/4
Sun	1/6	2/6	3/6	4/6	5/6	6/6	0	3/3	2/3	1/3
obscurity	1/7	2/7	3/7	4/7	5/7	6/7	7/7	0	2/2	1/2
Earth	1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8	0	1/1
Sun	1/9	2/9	3/9	4/9	5/9	6/9	7/9	8/9	9/9	0

Table 2. Reduced co-occurrence matrix, and resulting sentence feature vector

a. Reduced co-occurrence matrix

Centroid terms / *Aggregate context terms* / *Reduced context vectors* / *Aggregate weights*

	solar	eclipse	occurrence	Moon	passage	Earth	Sun	obscurity	*Aggregate weights*
solar	0	9/9	8/9	7/9	6/9	5/9+2/9	4/9+1/9	3/9	9/9+…+3/9
eclipse	1/1	0	8/8	7/8	6/8	5/8+2/8	4/8+1/8	3/8	1/1+…+3/8
occurrence	1/2	2/2	0	7/7	6/7	5/7+2/7	4/7+1/7	3/7	…
Moon	1/3	2/3	3/3	0	6/6	5/6+2/6	4/6+1/6	3/6	…
passage	1/4	2/4	3/4	4/4	0	5/5+2/5	4/5+1/5	3/5	…
Earth	1/5+1/8	2/5+2/8	3/5+3/8	4/5+4/8	5/5+5/8	0+6/8	4/4+7/8	3/4+8/8	…
Sun	1/6+1/9	2/6+2/9	3/6+3/9	4/6+4/9	5/6+5/9	6/6+6/9	0+7/9	3/3+8/9	…
obscurity	1/7	2/7	3/7	4/7	5/7	6/7+2/2	7/7+1/2	0	1/7+…+0

b. Sentence feature vector

/ Σ

	solar	eclipse	occurrence	Moon	passage	Earth	Sun	obscurity
Normalized weights	0.104	0.109	0.109	0.109	0.106	0.184	0.169	0.109

In brief, *k*-means [19, 20] attempts to divide data objects (e.g., sentences in our case) into non-overlapping subsets, i.e., the clusters, such that each sentence is in exactly one cluster, by maximizing intra-cluster similarity and minimizing inter-cluster similarity. It first chooses (randomly, or heuristically) *k* sentences as initial centroids, and computes one cluster around each centroid by associating sentences to the clusters sharing minimum distances with their centroids. The centroids are re-computed recursively, and the clusters are adjusted around them, until reaching convergence where the cluster centroids stabilize.

We utilize the cosine measure to compute sentence feature vector similarity, yet other similarity measures can be used (e.g., Jaccard, Dice, Euclidian, e.g., [18, 21]).

3.4. Summary Building

Our extractive summary building process consists in choosing the best representative sentence from each sentence cluster, and combining them together to form the output summary. Here, we consider multiple approaches (other approaches can be added according to the user's needs): i) *Longest Sentence* (LS) – from each cluster, the system extracts the sentence with the maximum number of terms (based on the assumption that the longest sentence is likely the most elaborate/descriptive of the cluster), ii) *Shortest Sentence* (SS) – from each cluster, the system extracts the sentence with the minimum number of terms (based on the assumption that the user is interested in the most concise information from the cluster), and iii) *Most Similar Sentence* (MSS) – from each cluster, the system evaluates the pairwise sentence similarities and chooses the sentence with the highest average similarity with all others (based on the assumption the sentence which shares the maximum amount of similarity would best reflect their information content). Once the representative sentences are identified, the system combines them to form the output summary, by placing them one after the other according to their relative order in the original text, to preserve logical content ordering.

The time complexity of our solution comes down to the complexity of the sentence clustering algorithm, requiring $O(N \times k \times |s|)$ where N is the size of the input text document in number of sentences, k is the number of clusters (i.e., the size of the output summary in number of sentences), and $|s|$ is the maximum size of a sentence in number of terms. It simplifies to $O(N \times |s|)$ since $k << N$.

4. Experimental Evaluation

4.1. Prototype Implementation

We have implemented our topical organization solution using the Python programming language, to test and evaluate its performance. We perform the data serialization using Python’s *PDFToText* library. We remove the stop-words and punctuations using Python’s *NLTK* library. Later we convert the remaining text to lower-case and we stem each word using the *NLTK Porter Stemmer*. We utilize the WordNet API to perform lemmatization.

4.2. Experimental Metrics

We make use of the compression ratio (*CR*) metric [22] to evaluate the compactness of the produced summaries, compared with the size of the initial text document:

$$CR = \frac{Length\ of\ Symmary}{Length\ of\ Full\ Text} \in [0,1] \quad \textbf{(2)}$$

We also make use of the precision (*PR*) and recall (*R*) metrics [18, 23] to evaluate the quality of the system produced summaries w.r.t. their human generated counterparts. High *precision* denotes that sentences are actually in the right summary. High *recall* means that very few sentences are not in the summary where they should have been. High *precision* and *recall*, and thus high *F-value* indicates excellent summarization quality.

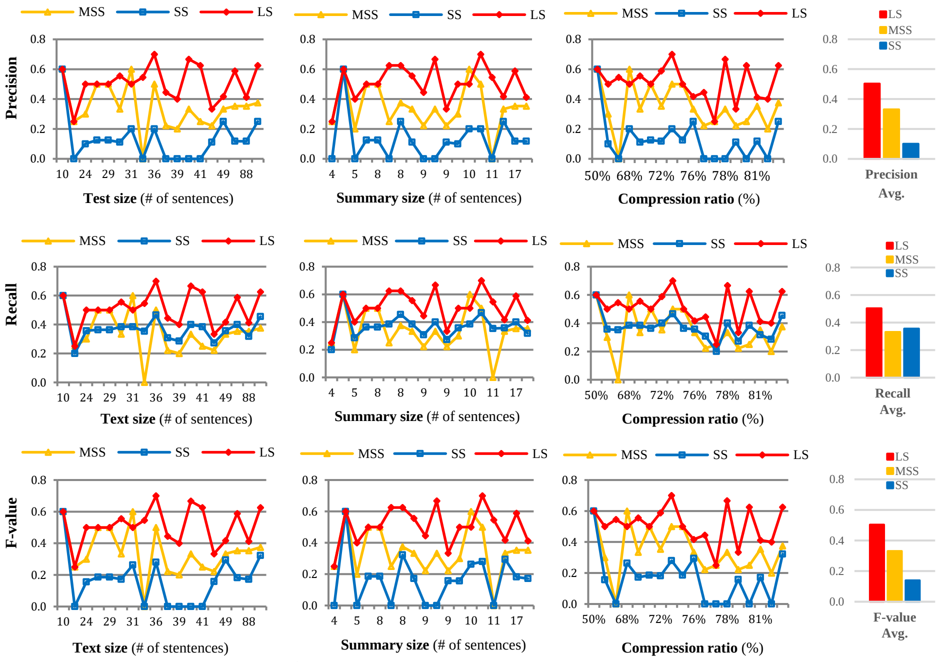
4.3. Experimental Data

We collected 18 news articles from the renounced online newspapers, consisting of an average 42 sentences and 855 terms per article[1]. We grouped the articles according to the desired (output) summary size, and requested the assistance of three human testers (graduates students) to generate the corresponding summaries. The testers were provided each an excel sheet that includes a header describing briefly the research and the experiment, and a body that includes an array containing as columns: i) reference number of source texts (articles), ii) summary size in number of sentences (e.g. k=2, k=4, ..., k=20), and iii) a link to the source articles. The testers completed the summarization tasks together, and compiled the summaries into the reference dataset in our experiments.

[1] Available online: https://bit.ly/3FiaMLu

4.4. Experimental Results

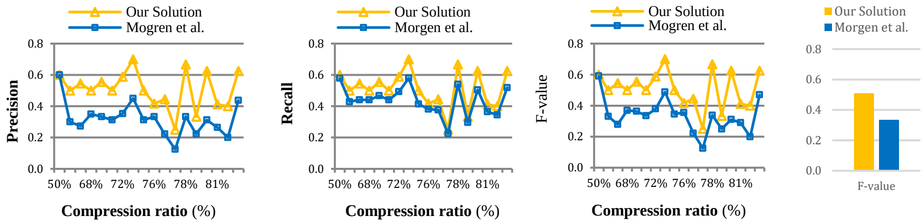
Fig. 3 shows the precision, recall, and f-value results obtained for our experimental dataset, considering the size of the input text, the size of the produced summary, the compression ratio, and each of the three representative options (longest sentence - LS, shortest sentence - SS, and most similar sentence - MSS).



a. Text size **b.** Summary size **c.** Compression ratio

Fig. 3. Precision, recall, and f-value results obtained for our experimental dataset

On the one hand, results in Fig 3 show that the quality levels of our summarization solution, considering all three precision, recall, and f-value metrics, remain more or less steady w.r.t. the varying size of the input text documents, the varying size of the output summaries, and the varying compression ratio. This shows that our solution produces consistent results regardless of text size, summary size, and compression ratio. On the other hand, results show that summarization quality is affected by the cluster representative selection approach, where the longest sentence (LS) method consistently produces the best quality levels (with average f-value = 0.51), followed by the most similar sentence (MSS) method (with average f-value = 0.33) , whereas the shortest sentence (SS) method usually ranks last (with average f-value = 0.14). Following our discussions with human testers, they usually consider the longest sentences to be the most informative and thus favor their presence in extractive summaries. Results also show that testers sometimes select sentences that are most related (i.e., similar) to others in the text. Nonetheless, testers very rarely allocate the shortest sentences in the summary, since they consider short sentences to be the least informative in the text.

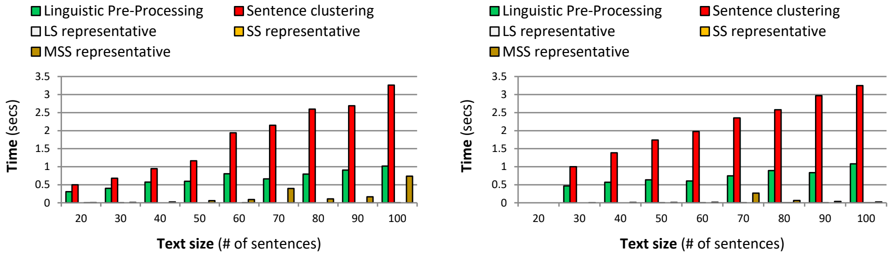


a. Precision **b.** Recall **c.** F-value

Fig. 4. Comparative evaluation

We compare our approach with an online extractive summarization solution by Morgen et al. [24, 25]. Fig. 4 shows the best results produced by our approach, i.e., considering the longest sentence (LS) method. Results show that our approach produces summaries which are more accurate, highlighting a higher average precision level = 0.51 (compared with 0.32 for Morgen et al.). Both solutions produce comparable recall levels, with a slight improvement in favor of our solution. Average f-value results highlight the quality of our approach, producing average 0.51 (compared with 0.34 for Morgen et al.).

Time performance results in Fig. 5 highlight the polynomial (almost linear) complexity of our approach, which comes down to the sentence clustering algorithm.



a. Size of summary (k) = 8 sentences **b.** Size of summary (k) = 20

Fig. 5. Time performance

5. Conclusion

This paper introduces a new solution for extractive text summarization, designed to process flat text documents. While most existing extractive solutions utilize term-based feature vectors combined with heuristic or linear optimization solutions, our solution performs sentence feature vector extraction, followed by sentence clustering using their feature vector similarity, and summary building based on the cluster representatives. Experimental results highlight our solution's quality and almost linear time complexity.

We are currently extending our experiments to consider a larger test dataset and to compare with multiple existing solutions. We are also investigating the selection process in the k-means algorithm and how to better consider the algorithm's convergence threshold in fine-tuning the summarization result, e.g., [19, 26]. Generating multiple representatives for each cluster and performing associative rule mining to select representatives [11] are also ongoing directions. We also plan to investigate encoder-decoder architectures, e.g., [27, 28], which have recently produced promising results.

References

[1] Khan A. and Salim N., *A Review On Abstractive Summarization Methods.* Journal of Theoretical and Applied Information Technology, 2014. Vol. 59 No.1.

[2] Tanaka H., et al., *Syntax-driven Sentence Revision for Broadcast News Summarization.* In Workshop on Language Generation and Summarization, 2009. pp. 39-47.

[3] Miller G. & Fellbaum C., *WordNet Then and Now.* Lang. Resources & Eval., 2007. 41(2): 209-214.

[4] Taddesse F.G., et al., *Relating RSS News/Items.* 9th International Conference on Web Engineering (ICWE'09), LNCS, 2009. pp. 44-452, doi: 10.1007/978-3-642-02818-2_36.

[5] Genest P. and Lapalme G., *Fully Abstractive Approach to Guided Summarization.* Annual Meeting of the Association for Computational Linguistics (ACL), 2012. pp. 354-358.

[6] Lau R., et al., *Toward a Fuzzy Domain Ontology Extraction Method for Adaptive e-Learning.* IEEE Transactions on Knowledge & Data Engineering, 2009. 21(6): 800-813.

[7] Özates S., et al., *Sentence Similarity based on Dependency Tree Kernels for Multi-document Summarization.* Inter. Conf. on Language Resources & Eval. (LREC), 2016.

[8] Shehab A.S. and Rafea A., *Performance Study on Extractive Text Summarization Using BERT Models.* Information journal, 2022. 13(2): 67 (2022).

[9] Cao S. and Yang Y., *DP-BERT: Dynamic Programming BERT for Text Summarization.* Inter. Conf. on Comput. Intel., Comm., & Business Analytics (CICBA), 2021. (2) 2021: 285-296.

[10] Aaditya M., et al., *Layer Freezing for Regulating Fine-tuning in BERT for Extractive Text Summarization.* Pacific Asia Conference on Information Systems (PACIS), 2021. 182.

[11] Haraty R. and Nasrallah R., *Indexing Arabic Texts using Association Rule Data Mining.* Library Hi Tech, 2019. 37(1): 101-117.

[12] Mansour N., et al., *An Auto-Indexing Method for Arabic Text.* Information Processing and Management journal, 2008. 44(4): 1538-1545.

[13] Mojrian M. and Mirroshandel S., *A Novel Extractive Multi-document Text Summarization System using Quantum-inspired Genetic Algorithm.* Expert Syst. Appl., 2012. 171: 114555 (2021).

[14] Srivastava A. et al., *Extractive Multi-document Text Summarization using Dolphin Swarm Optimization Approach* Multim. Tools Appl., 2021. 80(7): 11273-11290.

[15] Popescu M., et al., *A Highly Scalable Method for Extractive Text Summarization Using Convex Optimization.* Symmetry, 2021. 13(10): 1824.

[16] Kruengkrai C. and Jaruskulchai C., *Generic Text Summarization Using Local and Global Properties of Sentences.* Web Intelligence, 2003. pp. 201-206.

[17] Rani R. and Lobiyal D., *An Extractive Text Summarization Approach using Tagged-LDA based Topic Modeling.* Multim. Tools Appl., 2021. 80(3): 3275-3305 (2021).

[18] Tekli J., *An Overview of Cluster-based Image Search Result Organization: Background, Techniques, and Ongoing Challenges.* Knowl. Inf. Syst., 2022. 64(3): 589-642.

[19] Haraty R., et al., *An Enhanced k-Means Clustering Algorithm for Pattern Discovery in Healthcare Data.* Intelligent J. on Distributed Sensor Net., 2015. 11:615740:1-615740:11.

[20] Lloyd S., *Least Squares quantization in PCM.* IEEE Trans. Info. Theory, 1982. 28(2):129-137.

[21] Haraty R. and Hamdoun M., *Iterative Querying in Web-based Database Applications.* ACM Symposium on Applied Computing (SAC), 2002. pp. 458-462.

[22] Mridha M., et al., *A Survey of Automatic Text Summarization: Progress, Process and Challenges.* IEEE Access 2021. 9: 156043-156070 (2021).

[23] Tekli J. et al.., *Structural Similarity Evaluation between XML Documents and DTDs.* Inter. Conf. on Web Information Systems Engineering (WISE), 2007. pp. 196-211.

[24] Mogren O., et al., *Extractive Summarization by Aggregating Multiple Similarities.* Recent Advances in Natural Language Processing (RANLP), 2015. pp. 451-457.

[25] Kågebäck M., et al., *Extractive Summarization using Continuous Vector Space Models.* Workshop on Continuous Vector Space Models and their Compositionality (CVSC), 2014. pp. 31-39.

[26] Tekli J., et al., *(k, l)-Clustering for Transactional Data Streams Anonymization*. Information Security Practice and Experience, 2018. pp. 544-556.

[27] Maziad H., et al., *Preprocessing Techniques for End-To-End Trainable RNN-Based Conversational System.* Inter. Conference on Web Engineering (ICWE), 2021. pp. 255-270.

[28] Chakar J., et al., *Depthwise Separable Convolutions and Variational Dropout within the context of YOLOv3.* Inter. Symposium on Visual Computing (ISVC), 2020. pp. 107-120.