Information Extraction from Scientific Literature: Knowledge   
Management system

**Abstract**

In this study we propose a simple Knowledge Management System which can capture the percentage of similarity of a user provided research article with user given query. This system leverages the methods of text extraction from pdfs, text summarization using LSA and then calculation of cosine similarity score. We have also compared LSA model’s performance with a state-of-the-art model (BART) which is capable of text summarization. As known LSA is an extractive summarization technique, and it is used to detect the inherent relationship between a set of documents and terms contained in the document. In our case, we are using LSA to extract the semantic relationship of words of each section of the pdf and then fetch the top ranked sentences of each pdf section. Cosine similarity helps to measure the relevance between user given query and model generated summaries. Final output shows the combined results of each input pdf as form of an excel sheet and proves the effectiveness of proposed LSA model as well as BART. We have validated the quality of model generated summary with a reference summary using ROUGE, BLEU and METEOR scores. The model is built on some randomly sampled research articles of various domains.

**1. Introduction**  
With the growing demand of multiple factors, this is an era of continuous research and developments. Scientists and researchers are working hard to provide advanced technologies and methods to serve the well-being of a greater society. However, they struggle a lot when it comes to gathering relevant background information on any research topic. They need to spend a substantial amount of time to go through previous research papers for literature reviews and decide whether a particular paper is of their interest or not. These usually include short surveys, reviews, and conference proceedings along with scientific articles. Due to the scale of these articles and the amount of text in each one, there is an urgent need for a method where a human can easily access these articles selectively and very specific to their need and for understanding of the field, they desire to acquire knowledge via these articles. It is not possible for a human to go through all the academic articles available based on their field of interest (Zhi Hong et al.,2021) [[3](#three)]. Most of the existing paper recommendation systems are based out of complex methods like LSTM with attention mechanisms, Content based filtering (CBF) and Collaborative filtering (CF) etc. Our approach is different from above method as we tried to provide a section-wise similarity score of any pdf with respect to user query. This will help user to find not only the relevant pdf but also the relevant section within it.

The main contribution of this paper is summarized as follows: -

a. We try to do extract text from each section of each research paper provided by the user and then apply LSA for summarization of the text in each section. We use Bidirectional and Auto-Regressive Transformer from Hugging face as another method for summarization.

b. We then use Evaluation Metrics like ROUGE, METEOR and BLeU scores for checking how well the summarization has been done by the above algorithms by comparing to summaries made by human intervention.

c. We then try to match the query with text in each of the sections of the individual research paper and we provide an average similarity score for both BART as well as LSA as the primary summary methods for comparison.

d. We also match the query with the title and provide a separate similarity score for the same so the user can directly search for titles of research papers.

There are multiple advanced techniques for text summarization even if so many LLM models are being trained to serve the purpose, but we have utilized LSA in our experiment because of its efficacy and lesser implementation complexity than other models. Following sections will explain the detail understanding of our work as well as existing research that has been carried out in this area. We will also explain the methods used in three consecutive stages and the limitations we faced during experiments.

**3. Literature Review**

With the advancement of NLP field, the urge of finding out the hidden information from an

unstructured text data is increasing day by data. There is plethora of publications on each of text extraction, text summarization or cosine similarity measurement topic however, our focus is on building a system utilizing all above techniques. Researchers are also working on building recommendation system for scientific article search however, we have tried to create a simple one with help of very primitive NLP method like LSA which is quite comparable to LLMs in terms of efficiency. We may get to know about existing research works on this topic from this survey [[1](#one)]. A background study has also been done on how information is getting extracted from scientific articles [[2](#two),[3](#three),[4](#four)] and different text summarization methods [[5](#five),[6](#six),[7](#seven),[8](#eight)]. LSA method has been experimented widely for large text summarization [[5](#five),[7](#seven),[8](#eight)] however, we have tried to compare its performance in text summarization with respect to state-of-the-art LLM (Large Language Model). There are some methods like LDA, Map-Reduce [[9](#nine)] and graph methods [[10](#ten)] for text extraction but it comes with some limitations hence was not possible to use. We have discussed this in detail within following sections.

**4. Methodology**

Here, we aim to elucidate the diverse methodologies employed across various stages, detailing the inner functions and methods utilized within this research to accomplish the prescribed tasks effectively. This includes: -

1. Title extraction

2. Extracting the section headings and the text under each heading of the user provided research paper

3. Summarizing the text in each section using two methods namely: -

a. Latent Semantic Analysis (LSA)

b. Bidirectional and Auto-Regressive Transformers (BART)

4. Methods to evaluate the summaries outputted by the algorithms using 3 main evaluation metrics namely ROUGE, METEOR and BLEU scores.

5. Finding average similarity scores and outputting them to the user in the form of table in a CSV file format.

Tokenized sentence

Title

Similarity Calculation

parse

PDFs

Text

Query search

Similarity scores of individual papers

Section (Abstract, Introduction etc.)

Text Summarization using LSA

Chosen research papers



*Fig 1. Workflow diagram of our Knowledge Management Systems*

**4.1 Title Extraction**

Extracting the title of a research paper is super important as the title is a way to represent the entire paper in one line. We try to automate the extraction process of the title as our primary goal is to enhance the efficiency of handling research papers or documents by programmatically identifying and extracting titles from the PDF files. This therefore reduces the manual effort required for this task and thus helps in creating a pipeline.

The methods to try to extract the title mainly include the use of two Python Libraries namely PyPDF2 and PyMuPDF. The idea is to extract the text of the document and try to inspect the initial lines of the document for the potential title. If it is unable to detect the title of the PDF it resolves to alternate methods such as retrieving title from document metadata or taking the name of the PDF as the title as a last resort.

**4.2 Extracting the Headings of the Sections of the PDF**The first task in trying to extract the section headings is to gather information about the font size and font type of every single line in the PDF. Thus, this method is an efficient approach for extracting font related attributes helping in deeper analysis of the document structure and topography. This provides us valuable insight of the layout of the PDF and thus facilitates further formatting of the PDF.

The next step involves trying to extract the information of the font size and font text of the “Abstract” section of the PDF which is usually the very first section of the PDF. The purpose of this is to identify and filter text elements like the “Abstract” section of the PDF. Both font size and font name are used to filter text elements. These filtered items are then considered as headers for other sections within the document, providing further insights in the document’s structure organization and thereby helps us to extract each section headings of the PDF. During this procedure we also make sure to exclude the reference section from our PDF as we don’t consider it essential in term of matching the query with the text under the references section. Using the reference section in trying to match the query drastically reduces our average similarity score. After we extract headers of the section, further processing is done to refine and remove unwanted parts of the section headings which may include author names, acknowledgements, and other irrelevant section. Thus, this ensures that we are extracting only relevant section headers like Abstract, Conclusion etc. We make sure that we only identify section heading between “Abstract” and “Conclusion” excluding everything above and below these sections.

Once the headers are extracted, we store them, and we try to match these specific headers with the document provided by the user. Our method here tries to create a standardized manner to extract the Sections as well as the sub-sections of the PDF to make sure that we exclude the sub sections from the result and only keep the main headings of the document. By formatting header names in this manner, we ensure that we adhere to the original patter observed in the PDF’s.

The only pre-processing of the text in the entire document that we work up on is trying to remove the non-ascii characters and URLs from the code. This ensures better performance from the LSA as well as BART in summarizing each section and capturing the meaning of that section.

**4.2.1. Summarization Techniques**

Summarization stands as a pivotal component within our project, primarily driven by the imperative of reducing computational expenses and resource utilization. Notably, we have observed significantly enhanced similarity scores, particularly since user queries typically consist of only a few words. Focusing on determining similarity scores with respect to a summary of a section, rather than the entire text, yields notably improved and more accurate outputs. This approach aids in recommending the most relevant research papers to users based on their queries. ***a) Latent Sematic Analysis***

Latent Semantic Analysis is a statistical model that helps in the comparison of two pieces of text based on semantic similarity (Matthew S. McGlone, Peter W. Foltz, 1996) [[13](#thirteen)]. Here we will try to explain LSA briefly, more can be found in Deerwester, Dumais, Furnas, Landauer, & Harshman (1990) and Dumais (1990) [[14](#fourteen)].

The main assumption of LSA is that there are some underlying ways in which the words are used in documents (words, sentences, paragraphs) and LSA tries to understand this pattern.

The main working of LSA begins with forming a matrix of co-occurrences of each word in a document. Here we try to use Truncated Singular Value Decomposition which is a technique for eigenvector decomposition and factor analysis. The SVD scaling method dissects the word by document matrix into n orthogonal factors, typically ranging from 100 to 300, which can be combined linearly to approximate the original matrix. LSA does not represent documents and terms as vectors but instead it represents them as continuous values to each of the n orthogonal indexing dimensions obtained from SVD analysis. As the number of dimensions are significantly lower than the number of unique words in a document, words are not treated independently. If two words have similar context in two documents they will be represented as analogous vectors in the reduced LSA dimension representation. This is very useful because this helps in matching of two pieces of textual information even if they don’t share common words (Matthew S. McGlone, Peter W. Foltz, 1996) [[13](#thirteen)].

Geometrically, the analysis conducted by SVD can be understood as follows: The outcome of the SVD yields a k-dimensional vector space containing a vector representing each term and document. The positioning of term vectors reflects the correlations in their usage across documents, while document vectors reflect correlations in the terms used within them. Within this space, the cosine or dot product between vectors indicates their estimated semantic similarity. Hence, by establishing the vectors of two textual pieces, we can gauge their semantic resemblance. (Matthew S. McGlone, Peter W. Foltz, 1996) [[13](#thirteen)].

The application of LSA in our use case pertains to summarization of each section text based on LSA scores. We calculate LSA scores by summing the absolute value of LSA matrix along the rows along the rows (sentences). This yields a vector of scores, with each score corresponding to a sentence in a text. Then we group the sentences, and each sentence is paired with its respective section and LSA score. Thus, we summarize the text of each individual section by selecting the most informative sentences based on the LSA scores. By leveraging LSA scores and sentence grouping techniques, the code facilitates the extraction of key information from textual data, enabling concise and informative summaries for each section of the document. This summarization process aids in extracting meaningful insights and facilitating efficient understanding of document content, particularly in scenarios where manual summarization is impractical due to large volumes of text.

***b) Bidirectional and Auto-Regressive Transformer (BART)***

BART is a denoising autoencoder for pre-training sequence to sequence models. Its training process involves two main steps: (1) introducing noise to the text using a chosen noising function, and (2) training a model to reconstruct the original text. BART employs a conventional Transformer-based architecture commonly found in neural machine translation models. It features a standard sequence-to-sequence (seq2seq) or neural machine translation (NMT) architecture with a bidirectional encoder akin to BERT and a left-to-right decoder resembling GPT. This configuration implies that the encoder's attention mask is fully visible, like BERT, while the decoder's attention mask is causal, mirroring GPT2(Lewis et. al, 2019) [[12](#twelve)].

During the fine-tuning process for summarization tasks, BART undergoes additional training on datasets tailored for summarization. This fine-tuning aims to enhance BART's capability to produce abstractive summaries. In the summarization phase, the goal is to generate a summary Y based on the input sequence X. BART is fine-tuned to minimize the negative log-likelihood of the target summary given the input. BART's effectiveness stems from its transformer architecture, denoising autoencoder pre-training, and its proficiency in generating coherent and contextually relevant summaries during the fine-tuning phase. A comprehension of BART's self-attention mechanism, tokenization, and the specific objectives of pre-training and fine-tuning provides insights into its operation and excellence in text summarization tasks. (Sandeep Sharma, 2023) [[15](#fifteen)].

**4.3 Evaluating the Summaries given by LSA and BART**

Most state-of-the-art evaluation metrics rely on human generated summaries as reference summaries. There are other automated methods like Summary Score without Reference (SUSWIR) to assess the summaries generated by algorithms (Abdullah Al Foysal et. al, 2023) [[16](#sixteen)]. But we have decided to use the traditional method of using human generated summaries as reference summaries.

We have used 3 evaluation metrics for evaluating summaries of each section made by LSA and BART.

**4.3.1 Bilingual Evaluation Understudy (BLEU)**

Originally BLEU was a method developed for comparing a translation of text to other reference translation. But it can be modified to use to evaluate text generated for many Natural Language Processing tasks. For our project we will use it for comparing a generated summary to a reference summary. Summaries that perfectly match with each other will have a score of 1.0 and those that don’t match at all have a score of 0.0. (Jason Brownlee, 2019) [[17](#seventeen)].

The method operates by tallying matching n-grams in the generated summary against those in the reference summary(human-generated). Here, a 1-gram or unigram corresponds to each token, while a bigram comparison involves each word pair. The comparison disregards word order. The counting of matching n-grams is adjusted to consider the occurrence of words in the reference text, thereby discouraging the generation of summaries with an excess of common words. This adjustment is termed modified n-gram precision in the paper. While the score is initially designed for sentence comparison, a modified version that normalizes n-grams by their occurrence is also suggested for improved scoring of blocks containing multiple sentences. (Kishore Papineni et. al, 2002) [[18](#eighteen)].

Positives of BLEU score mainly include it being inexpensive to compute and it is language independent.

Some of the shortcomings of BLUE score are: -

1) It is more precision based than recall. So, it checks whether all the words in the generated summary are in the reference summary. It does not check if all the words in the reference summary are covered or not.

2) It does not consider the semantic meaning between words and the similarity between reference and generated summary.

3) Good comparisons between reference and generated summary rely upon clusters of words in generated summary closely match the clusters of words in reference summary, BLUE is not good in doing this. (MLNerds, 2021) [[19](#nineteen)].

**4.3.2 Recall Oriented Understudy for Gisting Evaluation (ROUGE)**

ROUGE is like BLEU primarily made for evaluating machine translations but is widely used as an evaluation metric for automated summarizations. We have used 2 types of ROUGE measurements for our evaluation namely ROUGE-N (ROUGE-1 and ROUGE-2) and ROUGE-L.

1) ROUGE-N – It measures the number of matching n-grams between the generated summary and the reference summary. We use ROUGE-1(unigram) and ROUGE-2(bigram) F1 measure for evaluation. (Fabio Chuisano, 2022) [[20](#twenty)].

2) ROUGE-L – It is based on the longest common subsequence (LCS) between the generated summary and the reference summary meaning that it will try to compare the longest sequence of words (not always consecutive, but still in order). The more the two text share a longer sequence of words the more the similarity between the generate and reference (human-generated summaries). Again, we use ROUGE-L F1 measure for evaluation. (Fabio Chuisano, 2022) [[20](#twenty)].

**4.3.3 Metric for Evaluation of Translation with Explicit Ordering (METEOR)**

It is another evaluation metric that helps check the alignment between the generated text and the reference text. It is the harmonic mean of unigram precision and recall, with recall weighted higher than precision. As we have already seen that BLEU is precision based and ROUGE is recall based, METEOR solves some of the issues in both BLEU and ROUGE.

METEOR is based on a very basic concept of unigram matching of generated text and reference text (human generated). Unigrams are matched mainly in their stemmed/ lemmatize forms, surface forms or meanings. Once the process of matching the generalized unigrams is done, METEOR computes a score based on the combination of unigram-precision, unigram-recall, and a measure of fragmentation that is designed in a way to directly capture how well ordered the matched words in the generated summary is to the reference summary.

**4.3.4 Application of our evaluation metrics**

We have tried to automate the process of accepting multiple reference summaries of the research papers in a csv format from the user and then comparing that to the generated summary and calculating the evaluation metrics.

**4.4 Measuring Similarity Scores between query and the document**

The application of similarity scores is very simple, it is trying to compare two pieces of text and measure how similar they are in terms of semantic relationship between them. Humans can understand and process words and their semantic meanings, but machines can’t do the same. Therefore, these words must be represented in the form of number. This is the idea behind embeddings. It is the conversion of a word, sentence, or even and entire document into a vector. This is how sentence similarity models work, they use these embeddings or convert words into embeddings themselves, thus capturing the semantic information in the input texts and calculate how close(similar) they are (Source – Huggingface.com).

It's crucial to acknowledge that similarity is subjective and strongly influenced by the domain and specific use case. For instance, the similarity between two cars can stem from various factors such as the manufacturer, colour, price range, or technical specifications like fuel type, wheelbase, and horsepower. Therefore, it's important to exercise caution when computing similarity across features that lack relevance or are unrelated to the problem at hand (Source – newscatcherapi.com) [[21](#twone)].

**4.4.1. Methods for creating embeddings of words to compare texts.**

In our project we have used two methods of creating embedding vectors: -

*a) Term Frequency-Inverse Document Frequency (TF-IDF): -*

TF-IDF embeddings are mainly an extension of One-Hot Encoding model. In One-Hot encoding vector is created with the size of the unique words in the corpora. Each unique word will have a unique feature and will be represented by 1’s with 0’s everywhere else.

Unlike in One-Hot Encoding models, in TF-IDF, instead of considering the frequency of words in one document, the frequency of words across the whole corpus is considered. Stop words like ‘is’, ’are’, ’and, is not given much significance. Nouns and Proper nouns that occur less frequently are given much more importance.

In mathematical terms, Term Frequency (TF) represents the frequency of a word appearing in a document divided by the total word count in the document. Inverse Document Frequency (IDF) is calculated as log(N/n), where N stands for the total document count and n represents the number of documents containing the term. The TF-IDF value for a word result from multiplying the term frequency by the inverse document frequency.

TF-IDF is a word embedding technique but for our project we have tried to create TF-IDF for sentence embedding.

Here's how TF-IDF works with sentences -

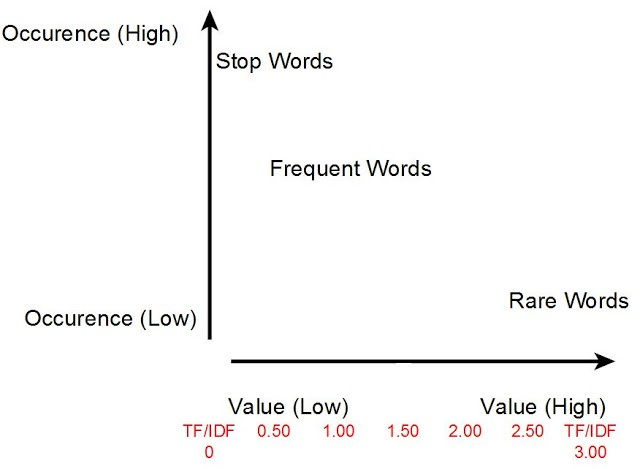
**TF-IDF for Words within Sentences**:

In its traditional form, TF-IDF calculates the importance of each word within a document based on its frequency (TF) in the document and its rarity (IDF) across the entire corpus. When applied to sentences, TF-IDF calculates the importance of each word within each sentence based on its frequency in that sentence and its rarity across all sentences in the corpus.

**Representing Sentences**:

After calculating TF-IDF scores for words within sentences, the scores for each word are typically aggregated to represent the entire sentence. One common method is to represent each sentence as a vector where each dimension corresponds to a unique word, and the value in each dimension is the TF-IDF score of the corresponding word in the sentence. Once sentences are represented as TF-IDF vectors, similarity between sentences can be measured using cosine similarity.

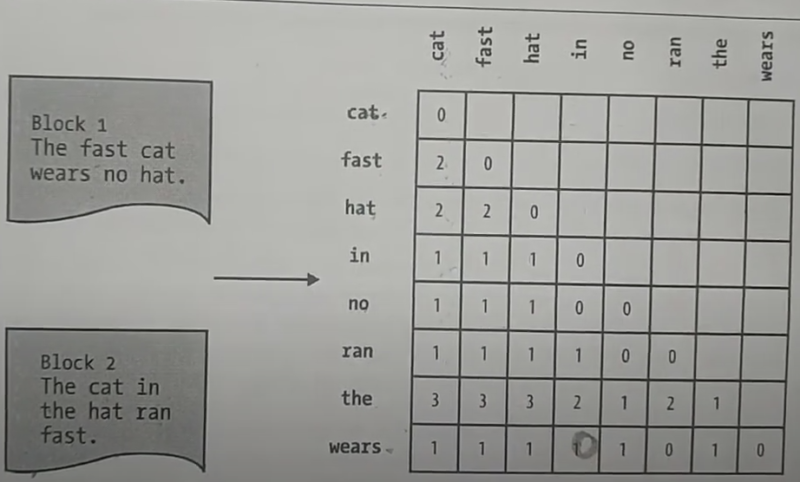
Despite TF-IDF vectors providing a modest enhancement over basic count vectorization, they retain a high level of dimensionality and fail to encapsulate semantic connections.



*Fig 2. TF-IDF (Source – newscatcherapi.com)*

*b) Global Vectors (GloVe): -*

GloVe, abbreviation for Global Vectors is an unsupervised learning system developed by Stanford University for creating word embeddings by aggregating global word co-occurrence matrices from a corpus. The co-occurrence matrix provides insights into the frequency of occurrence for a given pair of words appearing together. Each value in the matrix indicates the occurrence of a particular word pair within the context of the co-occurrence matrix (Mayank Goyal, 2023) [[22](#twwtwo)].



*Fig 3. Co-occurrence matrix (Source – codingninjas.com)*

The matrix resulted from merging the unique words from two datasets (blocks 1 and 2) into a unified matrix. Starting vertically from the word "cat," it appears once in block 1 and twice in block 2. Moving to the pair "cat-fast," it occurs once across both sets and twice in the provided corpus. Consider another pair, "cat-the," where 'the' accompanies 'cat' three times. The entire matrix follows this pattern. When calculating the probability ratio between two-word pairs, such as (cat/fast) = 1 and (cat/the) = 0.5, the result is 2, indicating that 'fast' holds more relevance than 'the.' This is the principal on which GloVe embedding works. (Mayank Goyal, 2023) [[22](#twwtwo)].

It is important to note that we have modified GloVe embedding method in a way where each vector does not represent each word but an entire sentence.

For more information refer ‘GloVe: Global Vectors for Word Representation’ by Jeffrey Pennington, Richard Socher, Christopher D. Manning [[23](#twwthree)].

**4.4.2 Distance Metric**

Cosine Similarity is the distance metric we have used. There are many other methods like Euclidean distance, Levenshtein distance but we have decided to go with the most popular method in cosine similarity.

Cosine similarity measures the similarity between two pieces of text based on the angle between the word vectors. In our case we have sentence tokenization hence we represent each sentence as a vector and not a word. Cosine similarity is usually used with TF-IDF vectors.

Cosine similarity assesses the likeness between two non-zero vectors within an inner product space. In the realm of document comparison, it's commonly utilized to gauge the resemblance between two documents depicted as vectors of word frequencies. The calculation of cosine similarity involves determining the cosine of the angle between the vectors. (Neri Van Otten, 2022) [[24](#tw4)].

To compute cosine similarity between two documents, initial steps involve constructing a vector representation for each document. Here, each dimension of the vector corresponds to a word in the document, with the dimension's value representing the frequency of that word in the document. Following this, the vectors are normalized to achieve a unit length. The cosine similarity is then calculated as the dot product of the two vectors divided by the product of their lengths. (Neri Van Otten, 2022) [[24](#tw4)].

The resultant cosine similarity value ranges from -1 to 1: -1 signifies entirely dissimilar documents, while 1 denotes identical documents. A value of 0 indicates that the two documents are orthogonal, indicating no similarity between them (Neri Van Otten, 2022) [[24](#tw4)].

**4.5 Displaying Output**

In our research study, we implemented a Python script to facilitate the storage of similarity scores into an Excel file. This script automates the process of recording and organizing similarity metrics obtained during our analysis. Specifically, our function orchestrates the storage procedure, ensuring accurate representation of similarity measures across different sections or summaries. This systematic approach not only enhances the organization of our research findings but also facilitates further analysis and interpretation of the data.

We are displaying two tables in our excel output:-

1) Similarity score of query to LSA summary using TF-IDF embedding method including all section similarity score and average similarity score

2) Similarity score of query to BART summary including all section similarity score and average similarity score.

We are also displaying the similarity score of the query to the title of the input research paper using TF-IDF embedding method.

**5. Results and Evaluations**

To explain the way, we have applied similarity scores for both LSA and BART we use the research paper “Using Convolutional Networks and Satellite Imagery to Identify Patterns in Urban Environments at a Large Scale” by Adrian Albert et. al, 2017 as our input data.

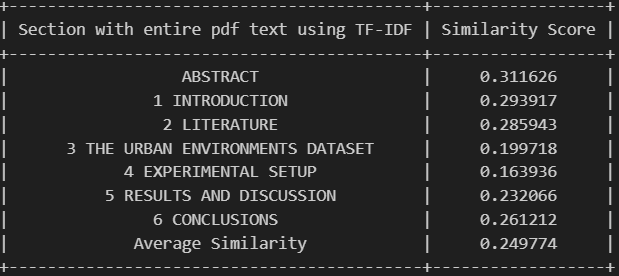
**5.1. Results**

**5.1.1. Similarity scores using LSA summary to query**

As mentioned previously we tried to use the above given embedding methods (sentence embedding) to convert the text summary from LSA and the query after doing three pre-processing steps on the text namely tokenization, lemmatization and stop word removal for better performance. The text is also converted to lower case before carrying out the pre-processing steps. Then we use cosine similarity as the distance metric to measure the similarity scores between the query text and the summarized text of each section of the document. Here we have tried to compare the query with summaries of LSA using TF-IDF and GloVe embedding, and we have tried to compare the query.

We have shown below the tables as the results of base (LSA) and challenger models’ (BART) final output.

1. *Scores for TF-IDF embedding:*



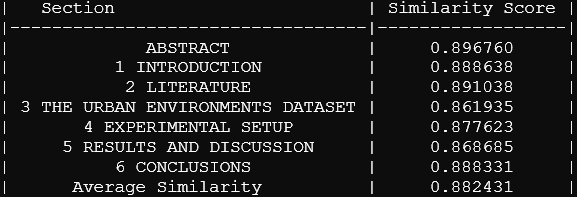
*Fig 4. Similarity scores of entire pdf extraction (without LSA).*

A black screen with white text

Description automatically generated

*Fig 5. Similarity scores of LSA applied part with Top-ranked sentences.*

1. *Scores for GloVe embedding*:



*Fig 6. Similarity scores entire pdf extraction (without LSA).*

A black screen with white text

Description automatically generated

*Fig 7: Similarity Scores of LSA applied part with Top-ranked sentences.*

1. *Similarity scores between pdf Title and user query*

Similarity score with Title 'using convolutional networks satellite imagery identify patterns urban environments large scale' using TF-IDF: 0.5773.

Similarity score with Title 'using convolutional networks satellite imagery identify patterns urban environments large scale' using GloVe: 0.8313.

**5.1.2. Similarity scores using BART summary to query**

While BART doesn't directly use pre-trained word embeddings, it learns contextual representations of tokens during pre-training and fine-tuning phases. These contextual representations capture the semantic and syntactic information of words and their context within sentences.

Below given the similarity scores after implementing pre-trained BART model.

A black screen with white text

Description automatically generated

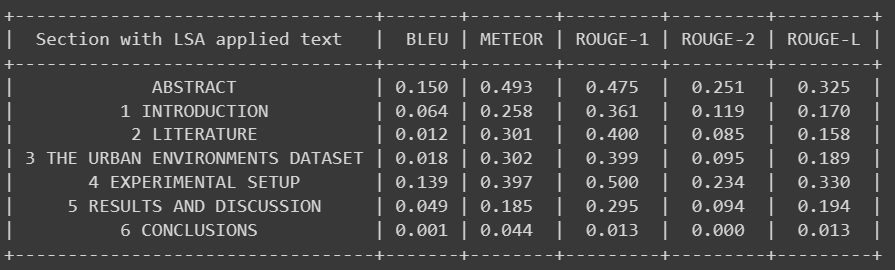
*Fig 8: Similarity Scores of user query vs LSA applied text using BART model*

**5.2. Evaluation of Results**

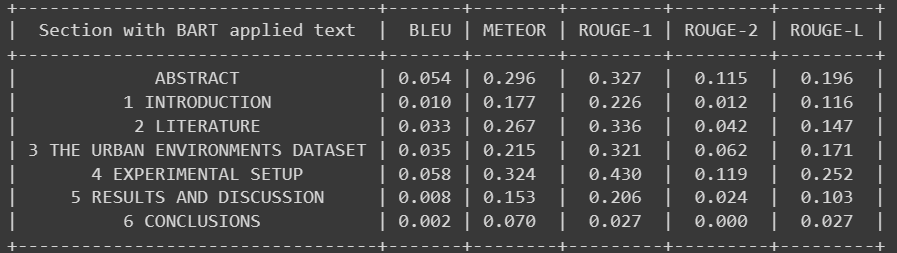
**5.2.1. Metrics to evaluate model generated summaries**

To evaluate the validity of above results we considered to validate the quality of summaries generated use different metrics like ROUGE, METEOR and BLEU scores (explained in previous section).

Due to unavailability of human expertise at this moment, we decided to rely on ‘Reference Summaries’ generated by ChatGPT for calculating the scores and below is the output.



*Fig 9. Evaluation metrics w.r.t. LSA model*



*Fig 10. Evaluation metrics w.r.t. BART model*

We employed a systematic approach to compare the performance of related research papers. The evaluation criteria included metrics such as accuracy, computational efficiency, robustness, and scalability. To ensure comprehensive analysis, we selected papers that address similar problem spaces or utilize comparable methodologies.

The selected research papers for comparison include Performance Analysis of Large Language Models for Medical Text Summarization (Amandeep Singh et. al,2023), Who Needs External References? Text Summarization Evaluation Using Original Documents (Abdullah Al Foysal et. al, 2023) which are widely recognized for their contributions to the field. These papers were chosen based on their relevance to our proposed solution and their impact within the research community.

The comparative analysis focused on various metrics to assess the performance of the selected research papers. These metrics were chosen to capture the key aspects of their effectiveness in addressing the problem domain. Upon evaluating the performance of the research papers, several noteworthy insights emerged. Each paper demonstrated strengths and weaknesses in different aspects of performance, highlighting the diversity of approaches within the domain. Through detailed analysis, we identified patterns and trends that shed light on the efficacy of existing methodologies.

*LSA performance*

The performance shown in the research paper of Who Needs External References? Text Summarization Evaluation Using Original Documents (Abdullah Al Foysal et. al, 2023) which uses LSA for text summarization, for raw data the ROUGE scores for BBC articles how a performance of 0.663 Rouge – 1 and Rouge – L and METEOR scores of 0.346. This is better over pre-processed data with scores of 0.639 for Rouge – 1 and Rouge – L. But for Meteor scores there is improvement by using pre-processed data with scores of 0.369.

*BART performance*

The performance shown in the research paper in include Performance Analysis of Large Language Models for Medical Text Summarization (Amandeep Singh et. al,2023) for BART we see for Rouge -1 scores of 0.373, Rouge – 2 scores of 0.160 and Rouge – L scores of 0.257. Meteor scores are 0.237 and BLEU scores are 0.076.

The comparison of performance in related research papers contributes to the advancement of knowledge within the field. By identifying areas of improvement and best practices, our findings offer valuable guidance for researchers and practitioners seeking to address similar challenges. It is important to acknowledge the limitations of our comparison study. Factors such as the selection of research papers, choice of evaluation metrics, and availability of data may have influenced the outcomes. Future research should consider addressing these limitations to ensure comprehensive analysis.

**5.3. Experimental Study**

As mentioned in literature review, we chose LSA over Graph method and LDA due to some reasons. First let’s get some brief introduction about these techniques:

**5.3.1. Graph-based method**

Graph-based methods, in the context of text analysis and NLP, involve textual data as graphs, where nodes typically represent entities and edges represent relationships between them. Such methods leverage the graph theory and algorithmic context to analyse, model and derive insights from underlying data structure. Generally, this type of method is widely used for finding out hidden semantic relationships between words, sentences or documents. Moreover, provide a flexible framework for integrating several information sources. Along with its scalability feature for large datasets this is also good fit for community detection and clustering.

*Limitations*: But we refrained from using graph-based method due to some drawbacks like –

1. Complexity – Building and analysing graphs from textual data is computationally quite expensive for large datasets.
2. Sparsity – Textual data often leads to sparse graphs, where many nodes have few connections or edges.
3. Interpretability – Graphs can capture complex relationships but interpreting and understanding the meaning of this relationships can be challenging.
4. These methods are also prone to overfit and derive noisy, erroneous conclusion within graphs.

**5.3.2. LDA**

LDA (Latent Dirichlet Allocation) is a topic-modelling technique used in NLP for finding underlying topic distribution of each document and the distribution of words for each topic from the observed word occurrences in the corpus.

*Limitations:*

1. Sparsity and Dimensionality – LDA produces sparse topic distributions, especially for short documents or when the data is too large.
2. Difficulty in choosing hyperparameters – LDA requires to specify the number of topics as hyperparameter and that is big challenge most of time because it needs tuning or cross-validation. Choosing incorrect number of topics can lead to a in-accurate results.
3. Complexity of inference – LDA inference involves iterative algorithms such as variational inference which can computationally expensive and slow for large datasets.

We skipped above methods because of their implementation complexity and computational inefficiency than LSA (Latent Semantic Analysis). Although, LSA is lagging in terms of interpretation, but we chose this as our base model due to its ease of handling and easy to understand features on any large text data.

**6. Conclusion and Discussions**

We have now successfully built a LSA driven KMS for searching scientific research articles.

It has been observed that, our proposed LSA model has noted the average similarity score as 0.031 (Fig. 5) and the pre-trained BART model has drawn the score as 0.034 (Fig. 10) So, we can consider our base model as good as an LLM in context of deriving the inherent textual relationships. Also, our LSA model seems to be performing better on non-processed text rather than processed text (especially while stop-words are removed). Hence, we kept the stop-words as it is within text extracted from pdf.

In terms of evaluation parameters also, both the models are maintaining balance in the output provided the reference summaries are not human generated.

In future, we will try to create a friendly application interface for end-users based on the current backend structure and will extend the scope of our input data from text to image with the help of computer vision techniques. Due to the restriction of resources provided by Hugging Face for using BART LLM we were only able to use the BART summarization feature with a restriction of 1000 tokens so that has affected the performance of the summaries provided by BART in a negative manner, something we will be looking up on for finding a solution for the same soon. Also, our system is now able to process a single pdf at a time so going forward our aim will be to enhance the efficiency of the system by adding the multiple input feature.

**7. References**

1. Christin Katharina Kreutz, Ralf Schenkel: Scientific paper recommendation systems: a literature review of recent publications. In: *International Journal on Digital Libraries* (2022).

2. Kiran Adnan, Rehan Akbar: An analytical study of information extraction from unstructured and multidimensional big data. In: *Journal of Big Data* (2019).

3. Zhi Hong, Logan Ward, Kyle Chard, Ben Blaiszik, Ian Foster: Challenges and Advances in Information Extraction from scientific Literature: A Review. In: *JOM: the journal of the Minerals, Metals & Materials Society 73(7612):1-18* (2018).

|  |
| --- |
| 4. Anna Marshalova, Elena Bruches, and Tatiana Batura: Automatic Aspect Extraction from Scientific Texts. [*arXiv:2310.04074*](https://arxiv.org/abs/2310.04074)***[cs.CL]*** (2023). |

5. Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei,Elizabeth D. Trippe, Juan B. Gutierrez: Text Summarization Techniques: A Brief Survey. [*arXiv:1707.02268*](https://arxiv.org/abs/1707.02268)***[cs.CL]*** (2017).

6. Daniel O. Cajueiro1,6,7, Arthur G. Nery1,7, Igor Tavares2, Ma´ısa K. De Melo3,7, Silvia A.

dos Reis4, Li Weigang5, and Victor R. R. Celestino3,7: A comprehensive review of automatic text summarization techniques: method, data, evaluation and coding. [*arXiv:2301.03403*](https://arxiv.org/abs/2301.03403)***[cs.CL]*** (2023).

7. Makbule Gulcin Ozsoy, Ilyas Cicekli, Ferda Nur Alpaslan: Text Summarization of Turkish Texts using Latent Semantic Analysis. In: *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference.*

8. Ben Hachey, Gabriel Murray & David Reitter: Dimensionality Reduction Aids Term Co-Occurrence Based Multi-Document Summarization. [*Proceedings of the Workshop on Task-Focused Summarization and Question Answering*](https://aclanthology.org/volumes/W06-07/) (2006).

9. N K Nagwani: Summarizing large text collection using topic modeling and clustering based on MapReduce framework. In: *Journal of Big Data* (2015).

10. Yamei Tu, Rui Qiu, Han-Wei Shen: SKG - A Versatile Information Retrieval and Analysis Framework for Academic Papers with Semantic Knowledge Graphs. [*arXiv:2306.04758*](https://arxiv.org/abs/2306.04758)*[cs.IR]*(2023).

11. Shrabanti Mandal, Girish Kumar Singh: LSA Based Text Summarization. In: *International Journal of Recent Technology and Engineering (IJRTE)*, July 2020.

12. Mike Lewis\*, Yinhan Liu\*, Naman Goyal\*, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, Facebook AI. [*arXiv:1910.13461*](https://arxiv.org/abs/1910.13461)*[cs.CL]* (2019)

13. Peter W. Foltz (1996): Latent Semantic Analysis for Text-Based Research.

14. Deerwester, Dumais, Furnas, Landauer, & Harshman and Dumais (1990): Indexing by Latent Semantic Analysis.

15. Sandeep Sharma (2023): Text Summarization with BART model.

16. Abdullah Al Foysal, Ronal Bock (2023) - Who Needs External References? Text Summarization Evaluation Using Original Documents.

17. Jason Brownlee (2023): A Gentle Introduction to Calculating the BLEU Score for Text in Python

18. Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu (2002): Bleu: A Method for Automatic Evaluation of Machine Translation.

19. MLNerds (2021): BLEU Score.

20. Fabio Chiusano (2022): Two minutes NLP — Learn the ROUGE metric by examples.

21. <https://www.newscatcherapi.com/blog/ultimate-guide-to-text-similarity-with-python>

22. Mayank Goyal (2023): Glove Embedding in NLP.

23. Jeffrey Pennington, Richard Socher, Christopher D. Manning: GloVe (2014): Global Vectors for Word Representation.

24. Neri Van Otten (2022): Top 7 Ways To Implement Document & Text Similarity In Python: NLTK, Scikit-learn, BERT, RoBERTa, FastText and PyTorch.