Pre2Mo-DMD: Pretrained Model runs on sentences ranked by Dynamic Mode Decomposition

Mona Singh
mona21053@iiitd.ac.in

Simran
simran21146@iiitd.ac.in

Smiti
smiti22072@iiitd.ac.in

Tarini Sharma
tarini19451@iiitd.ac.in

Abstract

Text summarization involves converting a document into a shorter summary without losing critical information. This shortens the time it takes to read a long document like a news article. It is seen that for a news article, it is focused on a single piece of news and the entire article just explains it with factual information and other details of the same, which might be of repetitive nature. The task of creating a summary for the articles would ensure that reader keeps up-to-date with the news by just reading the short summary and avoiding other less important information covered in the article. The summary also helps the reader to decide if it's read-worthy or not for him. This paper focuses on summarizing news articles using the ILSUM-2022 dataset. We propose a novel model **Pre2Mo-DMD** which captures the repetitive structure of news articles and throws away the less important information by using Dynamic Mode Decomposition so that it can be readily used by abstractive models with much higher performance. The model gives huge flexibility to use any abstractive model of choice as part of the architecture.

1 Problem definition

"I don't want a full report; just give me a summary of the results." This statement is heard so often nowadays because of the ample amount of information and content circulating over the internet that nobody has time to read the entire report or any article in one go. This calls for an Automatic Text Summarization system where the machine can understand the document and, based on the context, generates a summary for the user.

The goal is to filter out the wheat from the chaff and find the information that matters. This shortens the time it takes to read long material without losing critical information, making it easier to get a high-level understanding of the content. It also improves the effectiveness of indexing and enables commercial abstract services to increase the number of text documents they are able to process (Ano, 2020). Summarization can be applied when text has a lot of raw facts, for example, new articles and scientific papers. It is not useful when applied to texts where each sentence builds upon the previous, like stories and journals.

Text summarization can be divided into 2 approaches: **extractive** and **abstractive** approaches. Extractive approaches, as the name suggests, extract essential phrases from the document and combine them to form a condensed version of the original document. It does not create new phrases and only takes into account the words existing in the document to create the summary. These approaches might suffer from grammatical mistakes but are unlikely to change the meaning of the text as it simply performs phrase extraction (Vic, 2021). Abstractive approaches, on the other hand, generate phrases to capture the meaning of the document. It tries to understand the meaning of the whole text, creates and combines new phrases, and adds the most important facts found in the text. However, these approaches are computationally more expensive and require a larger dataset as compared to extractive approaches. End-to-end training help in generating grammatically correct summaries. However, these approaches might create new text that changes the meaning of the original document (Vic, 2021).

On the basis of the purpose, summarization can also be divided into 3 categories: generic, domain-specific, and query-based (Ano, 2020). In generic text summarization, the model does not assume anything about the text domain, which is in contrast to domain-specific where the model utilizes domain-specific knowledge to generate better summaries (for example, generating summaries of specific domain research papers). In query-based summarization, the summary only contains information that answers queries regarding the input document.

Furthermore, summarization can be classified into 2 classes based on the input type: single document (where input length is short) and multi-document (dealing with multiple documents to generate summary and the input can be arbitrarily long) (Ano, 2020).

2 Related works

In (Luhn, 1958), the authors assigned a score to each sentence depending on the word frequency and extracted the sentence with the highest score. Their main idea was that high-frequency words determine the important concepts of the document. In (Edmundson, 1969), the author observed that initial sentences of a document usually contain the topic information and incorporate the concept of sentence location, title, heading, and cue phrases along with word frequency to extract a summary. (Zhang and Li, 2009) use a cluster-based approach where they first cluster document sentences based on semantic distance using k means clustering and then calculate the accumulative sentence similarity between the sentences for each cluster using the multi-features combination method. Then based on some extraction rules, summary sentences are extracted.

In (Rush et al., 2015), the authors used a generation algorithm with an attention-equipped neural network to generate summaries. In (Chopra et al., 2016), the authors use conditional RNN as their model where RNN is conditioned on a convolutional attention-based encoder so that decoder can attend to important words during summary generation. (Hu et al., 2015) used RNN to generate a summary in the Chinese language on a newly curated dataset of Large-scale Chinese Short Text Summarization, which was based on a microblogging site. In (Nallapati et al., 2016), the authors use an attention-based encoder-decoder RNN architecture where they utilize a large vocabulary trick. A feature-rich encoder is used to capture important concepts in the document. To deal with rare words, the authors use a switching generator-pointer mechanism and also use hierarchical attention to capture important sentences for long documents. (Lebanoff et al., 2020) showcased the possibility of using a cascade architecture for neural text summarization. The paper uses content selection and surface realization jointly to generate summaries.

(Zhang et al., 2020) used PEGASUS for performing abstractive summarization. PEGASUS proposed a new self-supervised pre-training objective for abstractive summarization and gap-sentence generation. It selects and masks the whole sentences from documents and concatenates those gap sentences as pseudo-summary.

(Zhao et al., 2022) introduced a sequence likelihood calibration (SLiC) with the conditional language model, which calibrates the likelihood of model-generated sequences to better align with reference sequences in the model's latent space. This significantly improves the decoding candidates' quality, regardless of the decoding method. Also, SLiC presents alternative ways to improve quality with limited training and inference budgets.

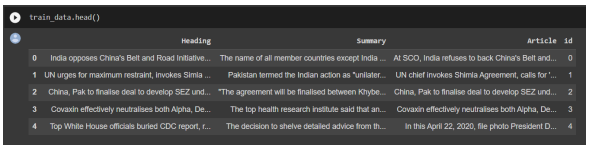


Figure 1: Original Train dataset

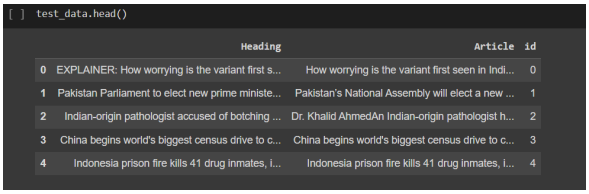


Figure 2: Original Test dataset

3 Dataset

3.1 ILSUM 2022 Dataset

The train dataset has 4 attributes: Heading, Summary, Article and ID and test dataset has 3: Heading, Article and ID. Train dataset has 10,052 data samples and test dataset has 2513 samples. The first 5 data samples for the train dataset is shown in Figure 1 and for test dataset is shown in Figure 2.

3.2 Exploratory Data Analysis

None of the attributes have a Null value for any data sample, be it a train or test dataset. To decide the encoder and decoder length, the authors plotted the length of the headline, article, and summary vs. frequency graphs for the train set (Figure 4, 5, 6). A similar analysis was done for the test set, where the authors plotted the length of the headline and article vs. frequency graphs (Figure 8, 7). To reduce the influence of noise, it is better not to choose the maximum encoder (or decoder) length as the maximum length of the article (or summary).

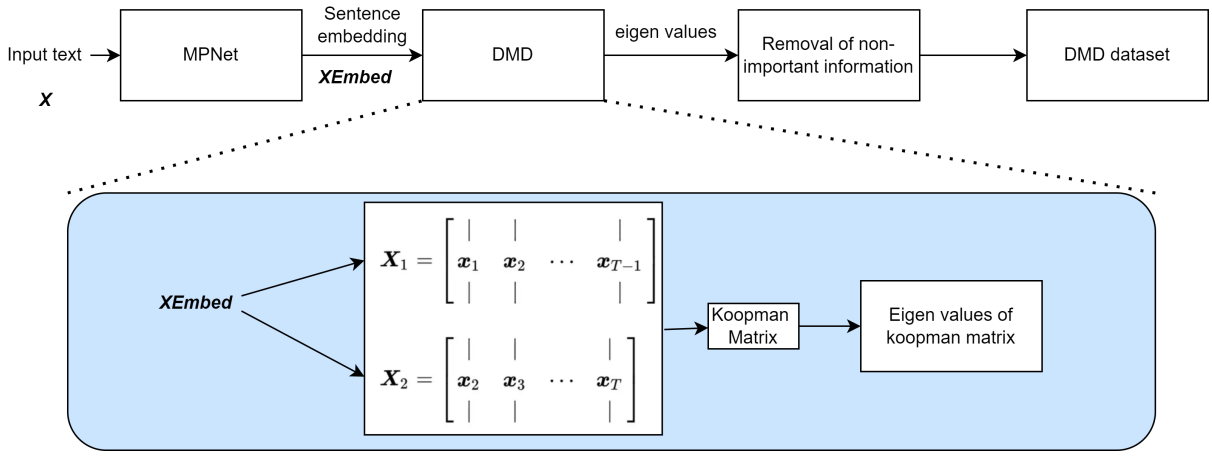


Figure 3: Pre2Mo-DMD's Steps to calculate eigen values

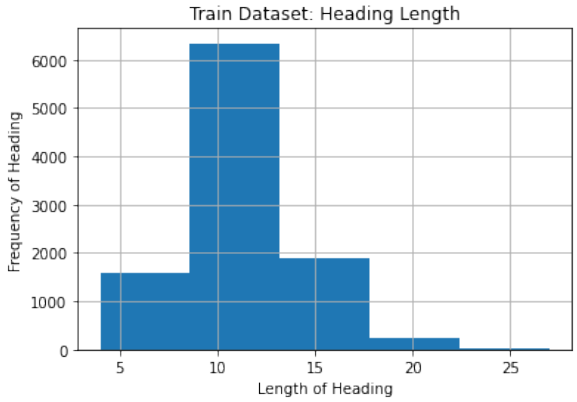


Figure 4: Train dataset: heading length vs. frequency

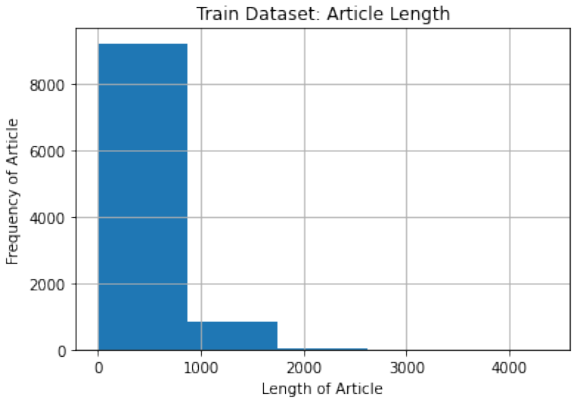


Figure 5: Train dataset: article length vs. frequency

Instead, fix the maximum heading length as 18, article length as 1500, and summary length as 56, which covered a majority of the data samples (Figure 9, 10).

3.3 Preprocessing

On manual analysis of the training dataset, we found out that there were lots of Unicode characters in the text. These were detected and removed using the regex : `[^\x00-\x7F]+`. Further analysis revealed that some data samples also contained javascript code which was removed using regex: `r\/\*`. Some articles also had an "also read" section which contained links to similar articles, these were removed using regex: `r"also read"`. When performing tokenization, it was observed that some sentences did not contain a space after ".", which hindered effective tokenization using NLTK. To handle this issue, first, we checked if the character after the full stop was not a space or a digit (since we should

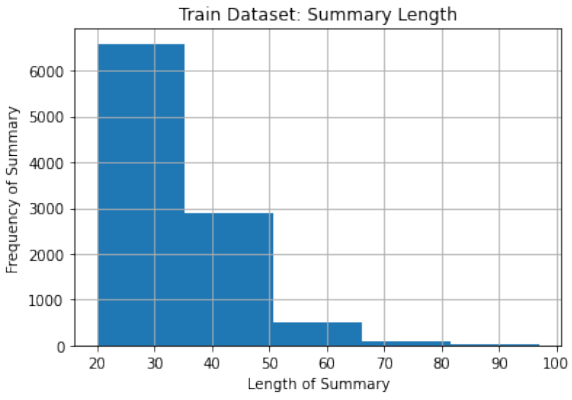


Figure 6: Train dataset: summary length vs. frequency

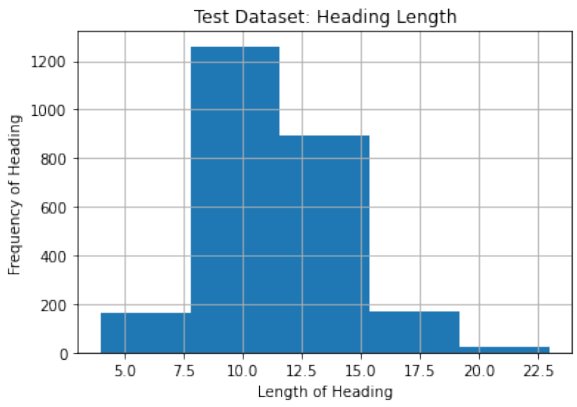


Figure 7: Test dataset: article length vs. frequency

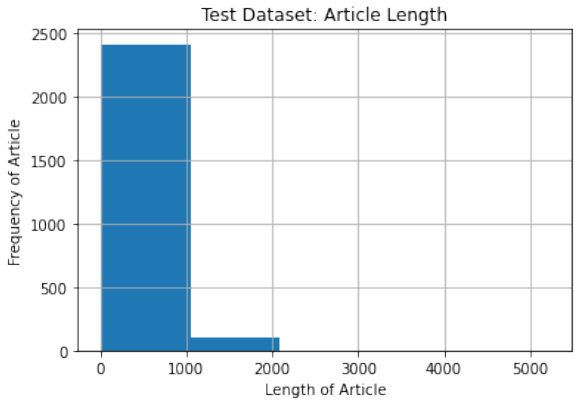
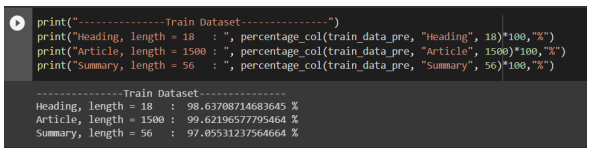


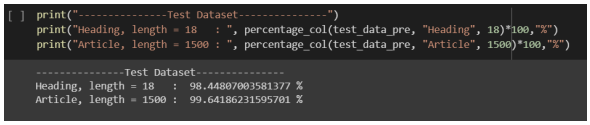
Figure 8: Test dataset: heading length vs. frequency



```
print("---------------Train Dataset--------------")
print("Heading, length = 18   : ", percentage_col(train_data_pre, "Heading", 18)*100,"%")
print("Article, length = 1500 : ", percentage_col(train_data_pre, "Article", 1500)*100,"%")
print("Summary, length = 56   : ", percentage_col(train_data_pre, "Summary", 56)*100,"%")

---------------Train Dataset--------------
Heading, length = 18   :  98.63708714683645 %
Article, length = 1500 :  99.62196577795464 %
Summary, length = 56   :  97.05531237564664 %
```

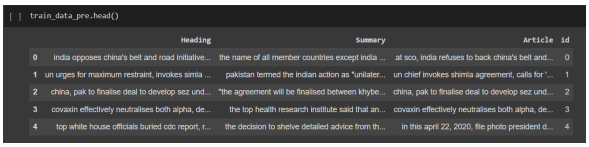
Figure 9: Train dataset: heading, article, summary length vs. data samples with length <= fixed maximum length



```
[ ] print("---------------Test Dataset--------------")
    print("Heading, length = 18   : ", percentage_col(test_data_pre, "Heading", 18)*100,"%")
    print("Article, length = 1500 : ", percentage_col(test_data_pre, "Article", 1500)*100,"%")

    ---------------Test Dataset--------------
    Heading, length = 18   :  98.44807003581377 %
    Article, length = 1500 :  99.64186231595701 %
```

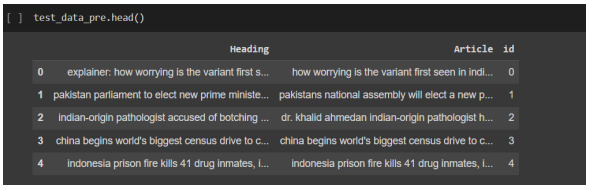
Figure 10: Test dataset: heading, article length vs. data samples with length <= fixed maximum length



```
[ ] train_data_pre.head()
```

	Heading	Summary	Article	id
0	india opposes china's belt and road initiative...	the name of all member countries except india ...	at sco, india refuses to back china's belt and...	0
1	un urges for maximum restraint, invokes simla ...	pakistan termed the indian action as "unilater...	un chief invokes shimla agreement, calls for '...	1
2	china, pak to finalise deal to develop sez und...	"the agreement will be finalised between khybe...	china, pak to finalise deal to develop sez und...	2
3	covaxin effectively neutralises both alpha, de...	the top health research institute said that an...	covaxin effectively neutralises both alpha, de...	3
4	top white house officials buried cdc report, r...	the decision to shelve detailed advice from th...	in this april 22, 2020, file photo president d...	4

Figure 11: Preprocessed Train dataset



```
[ ] test_data_pre.head()
```

	Heading	Article	id
0	explainer: how worrying is the variant first s...	how worrying is the variant first seen in indi...	0
1	pakistan parliament to elect new prime ministe...	pakistans national assembly will elect a new p...	1
2	indian-origin pathologist accused of botching ...	dr. khalid ahmedan indian-origin pathologist h...	2
3	china begins world's biggest census drive to c...	china begins world's biggest census drive to c...	3
4	indonesia prison fire kills 41 drug inmates, i...	indonesia prison fire kills 41 drug inmates, i...	4

Figure 12: Preprocessed Test dataset

not insert space in decimal numbers). If not, then we added a space after the full stop using: `re.sub(r"\.([^0-9 ])",r". \1",txt)` The following preprocessing steps were followed for both train and test data samples:

- converting to lowercase
- removing Unicode characters
- removing javascript code
- removing text after "also read" section
- inserting space after "." for efficient tokenization

Datasets after preprocessing are shown in Figure 11 and 12 for train and test datasets respectively.

Train-Validation Split Strategy: For splitting the train set into train and validation set, the authors computed the ROGUE score of summary with the article, then ordered the samples in increasing order of ROGUE and picked the lowest 10% samples as the validation set. This ensures that the summary is not just comprised of phrases from the article; the model has to understand the context fully and then generate a summary.

4 Baselines

4.1 Models

The authors trained extractive and abstractive models on the ILSUM 2022 English dataset.

4.1.1 Extractive Models

There are 3 different approaches tried which are listed as follows:

- extracting the sentence which has highest rouge score with the document
- extracting top 10% sentences which have highest rouge score with the document

- bert extractive summarization which combines TextRank and Lead3 and bert understands the context

4.1.2 Abstractive Models

The authors relied on 2 different approaches

- T5-small
  `t5-small`
- T5 fine tuned on news dataset
  `t5-base-finetuned-summarize-news`

5 Methodology/ Proposed Architecture

For a news article, it is observed that it is focused on a single piece of news and the entire article just explains it with factual information and other details of the same, which might be of repetitive nature. We propose a novel model **Pre2Mo-DMD** which captures the repetitive structure of news articles and throws away the less important information by using Dynamic Mode Decomposition so that it can be readily used by abstractive models with much higher performance.

5.1 Module 1: MPNet

Initially, the sentences are converted to sentence embeddings using MPNet: Masked and Permuted Pre-training for Language Understanding. This sentence transformer is used because it scores highest till date for semantic similarity. The embeddings are then feeded into DMD module.

5.2 Module 2: Dynamic Mode Decomposition (DMD)

Dynamic Mode Decomposition (DMD) is a powerful tool for extracting spatial and temporal patterns from multi-dimensional time series. The text is also a kind of time series data where the words come together in a sequence to convey some information. The most important tools in DMD are singular value decomposition and eigenvalue decomposition. To compute the Koopman matrix, there are only two steps that should follow: Perform singular value decomposition on the data matrix X1, Implement a truncated singular value decomposition with a certain predefined rank r, and compute the Koopman matrix. Next step is to perform eigenvalue decomposition on the Koopman matrix. These eigen values are returned which helps in removing less important information from articles. After this step new dataset called DMD dataset is generated which can be fed to any abstractive model. Figure 3 shows the steps performed to generate the eigen values.

The pseudo-code for Dynamic Mode Decomposition (DMD)

```
Algo DMD(data, r):

    ## Build data matrices
    X1 = data[:, : -1]
    X2 = data[:, 1 :]
    ## Perform SVD on X1
    ## in full_matrices = False
    u, s, v = np.linalg.svd(X1)
    ## Compute the Koopman matrix
    A_tilde = u[:, : r].conj().T
              @ X2 @
              v[: r, :].conj().T
              * np.reciprocal(s[: r])
    ## eigenvalue decomposition
    Phi, Q = np.linalg.eig(A_tilde)

    return Phi
```

5.3 Module 3: Pretrained Model

The model gives huge flexibility to use any abstractive model of choice as part of the architecture. The user can use the abstractive model which is shown to give good results and just use Pre2Mo-DMD as a black box which gives as output as better dataset which is sure to perform well on any model. Figure 13 showcases the architecture for the novel model: Pre2Mo-DMD. This model can also be extended to other text summarization domains where there is one major information and other details just follow that one piece of information.

6 Experimental Results

The experimental details are shown below.

6.1 Evaluation Metrics

The evaluation metric used is the average of ROUGE-1 and ROUGE-2 scores.

6.2 Results

The results for both extractive and abstractive models are shown in Table 1. These results are shown for 25% of the test dataset.

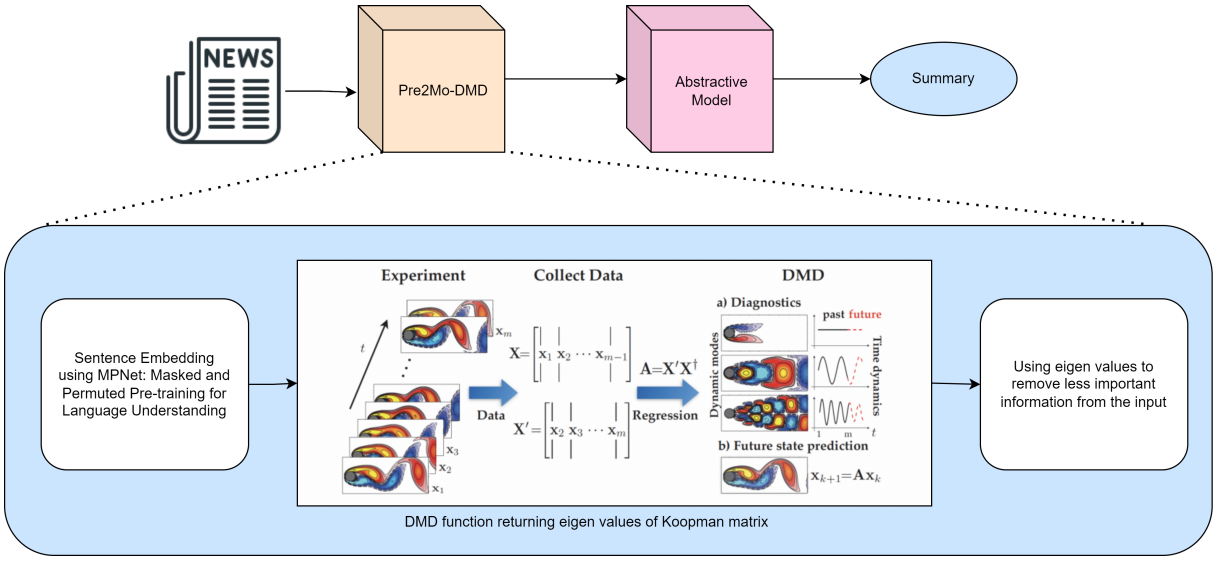


Figure 13: Proposed Model: Pre2Mo-DMD

Table 1: Results on baselines and novel model (Average of rouge 1 and rouge 2)

Models	Rouge Score	Model type
sentence which has highest rouge score with the document	0.365	Extractive
top 10% sentences which have highest rouge score with the document	0.365	Extractive
bert extractive summarization which combines TextRank and Lead3 and bert understands the context	0.352	Extractive
t5-small	0.357	Abstractive
T5 fine tuned on news dataset	0.378	Abstractive
Pre2Mo-DMD with T5 fine-tuned on news	**0.418**	Extractive & Abstractive

7 Analysis

Extractive summarization is able to give good results despite its naive idea. This is because of the fact that in news articles, the most important information lies in the title of the news and the top 3 sentences of the article itself. To further enhance the rouge, there is a need to incorporate abstractive models to generate summaries. Abstractive models try to understand the context and present the most important information of the article, but they may end up having lower rouge score due to obvious reasons. Pre2Mo-DMD uses the best of both worlds and hence gets the highest rouge and is able to beat the baselines.

8 Contribution of members

Table 2 depicts the contribution of the members.

References

2020. Text summarization using nlp. https://medium.com/analytics-vidhya/text-summarization-using-nlp-3e85ad0c6349.

2021. Neural extractive summarization with bert. https://victordibia.com/blog/extractive-summarization/.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Table 2: Contribution of members

Name	Contribution
Mona	• data preprocessing • abstractive: t5-finetuned-on-news-dataset • Novelty formulation of MPNet module in Pre2Mo-DMD • Coding of MPNet module in Pre2Mo-DMD
Simran	• data preprocessing • abstractive: t5-small • Novelty formulation of DMD module in Pre2Mo-DMD • Coding of DMD module in Pre2Mo-DMD
Smiti	• EDA • extractive: sentence with highest rouge score with document • bert extractive
Tarini	• EDA • extractive: top 10% sentence with highest rouge score with document • bert extractive

H. P. Edmundson. 1969. New methods in automatic extracting. *J. ACM*, 16(2):264–285.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. LCSTS: A large scale Chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1967–1972, Lisbon, Portugal. Association for Computational Linguistics.

Logan Lebanoff, Franck Dernoncourt, Doo Soon Kim, Walter Chang, and Fei Liu. 2020. A cascade approach to neural abstractive summarization with content selection and fusion. *arXiv preprint arXiv:2010.03722*.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Pei-ying Zhang and Cun-he Li. 2009. Automatic text summarization based on sentences clustering and extraction. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 167–170.

Yao Zhao, Misha Khalman, Rishabh Joshi, Shashi Narayan, Mohammad Saleh, and Peter J Liu. 2022. Calibrating sequence likelihood improves conditional language generation. *arXiv preprint arXiv:2210.00045*.