Finding Proxy For Human Evaluation
Re-evaluating the evaluation of news summarization

by

TARANG J. RANPARA
202011057

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

July, 2022

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

Tarang J. Ranpara

Certificate

This is to certify that the thesis work entitled "Finding Proxy For Human Evaluation: Re-evaluating the evaluation of news summarization" has been carried out by Tarang Ranpara for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.

Dr. Prasenjit Majumder
Thesis Supervisor

Acknowledgments

"If we knew what we were doing, it would not be called research, would it?"

- Albert Einstein

I express my deepest gratitude to Professor Prasenjit Majumder for being an ideal teacher, mentor, and thesis supervisor, offering advice and encouragement with insights into the field of information retrieval and natural language processing. He has taught me the methodology to carry out the research and to present research works as clearly as possible. His dynamism and vision have deeply inspired me. I'm proud of and grateful for my time working with Prof. Prasenjit at the information retrieval and language processing lab, DA-IICT.

I am also grateful to Surupendu Gangopadhyay, who, along with Prof. Prasenjit, taught the courses on Information retrieval and Natural language processing. This thesis builds on work done in those courses, and many of its ideas were developed during insightful discussions with a fellow labmate, Saran Pandian.

I would be remiss in not mentioning my friends Khushali Ratanghayara, Devansh Choudaha, Darshil Patel, Shivangi Gajjar, Dhyanil Mehta, Kishan Vaishnani, and Meet Shah, who stuck with me during thick and thin and made this journey a memorable one.

Last but not least, I'm thankful to my parents and siblings for constantly believing in me and pushing me when needed!

Contents

Abstract

Engaging human annotators to evaluate every summary in a content summarization system is not feasible. Automatic evaluation metrics act as a proxy for human evaluation. A high correlation with human evaluation determines the effectiveness of a given metric.

This thesis compares 40 different evaluation metrics with human judgments in terms of correlation and investigates whether the contextual similarity-based metrics are better than lexical overlap-based metrics, i.e., ROUGE score. The comparison shows that contextual similarity-based metrics have a high correlation with human judgments than lexical overlap-based metrics. Thus, such metrics can act as a good proxy for human judgment.

Keywords: News Summarization, Evaluation, Lexical overlap, Contextual Similarity, ROUGE, Transformers, word2vec

List of Principal Symbols and Acronyms

Annotator Reviewer

GRU Gated Recurrent Unit

Human judgement Human Review

LSTM Long Short-Term Memory

NLP Natural Language Processing

RNN Recurrent Neural Networks

List of Tables

List of Figures

CHAPTER 1
Introduction

We live in a fast-paced world where enormous amounts of data are being generated every minute; we are experiencing information overload. In order to make sense of it and extract knowledge from it, the process of summarization becomes critical. We now see summaries of news articles, books, and research papers/articles being supplemented with actual content. Search engines provide short snippets of summary along with search results; news sites provide a summary at the start of the article. Generally, readers tend to read summaries first, and they continue reading the main content if they find it interesting.

The short form of content is becoming increasingly popular because of the observed decrease in attention span [3]. Thus, for content publishers like news sites, social media sites, and info repositories(like Wikipedia), generating content summaries becomes essential to stay competitive.

Radev et al. define a summary as "A summary can be loosely defined as a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that." [13]. This definition highlights three main points about a summary:

1. A summary can be created from one or many documents.
2. A summary should preserve important information from the source document(s). It should present the "main idea" of its source(s).
3. A summary should be short.

1.1 Types of Summarization Algorithms

Method-wise, summarization algorithms fall into two types: Extractive summarization and Abstractive Summarization.

1.1.1 Extractive Summarization

The extractive summarization tries to select important sentences from the source document(s). The extractive Summarization algorithms are generally straightforward, they often come down to a binary classification problem about whether to include a sentence in a summary or not. Analogically, the process is more like highlighting the essential parts of the text. Few of the classic Extractive summariation algorithms include LexRank and TexRank [2].

1.1.2 Abstractive Summarization

In contrast with Extractive summarization, Abstractive summarization tries to grasp the idea of the source document(s) and generate a novel summary. It resembles more with how a human would summarize a document. Abstractive summarization algorithms are comparatively complex and often posed as a seq-to-seq problem. Traditionally, seq-to-seq problems are solved by Recurrence-based architectures like LSTMs, and GRUs, where the encoder tries to grasp the main idea of the document. At the same time, on the decoder side, we run a classification problem over a whole vocabulary to decide which word to produce on every timestep.

Since the inception of transformers [18], they have almost replaced recurrence-based architectures like RNN, LSTM, and GRU by outperforming them on multiple NLP tasks, including translation, summarization, classification, and Question Answering. One of the critical advantages that transformers offer over recurrence-based architectures is that we can parallelize the computation. With advanced language understanding capabilities, ease of deployment, and tools and resources from major cloud platforms, transformers dominate content summarization systems.

1.2 Evaluation of Summarization

In a content summarization system, it is essential to monitor the quality of summaries produced continuously. While assessing the quality of summaries, reviewers consider qualitative measures like grammatical correctness, the flow of information, conciseness, exhaustiveness, and domain suitability [16]. However, it is impossible to have humans review every summary produced because of time and resource constraints. We need automatic evaluation metrics that can function without human intervention and act as a proxy for human judgments. To judge how good a given evaluation metric is, we compute its correlation with human judgments; the higher the correlation, the better. Automatic Evaluation metrics are broadly classified into two types:

1. Lexical Overlap

2. Contextual Similarity

1.2.1 Lexical Overlap

ROUGE [6] is a lexical-overlap-based metric, traditionally the most common metric used for summarization. It captures N-gram lexical overlap between candidate summary and reference summary. Its popularity can be attributed to the fact that it is a statistical metric independent of the type of data it is applied to.

1.2.2 Contextual Similarity

In contrast with ROUGE, contextual similarity-based metrics compute the similarity between the underlying meaning of the candidate and the reference summary. For capturing the meaning, we project both candidate and reference summary in contextual embedding space using various methods like BertScore [21], word2vec [12], Glove Embeddings [11], and Transformer models like BERT [1], ROBERTA [8]. We then compute the similarity between both vectors using cosine-similarity or word-movers-similarity.

Capturing the similarity between underlying meanings sounds like a fantastic idea, but it comes with its cost. Capturing the meaning of a document often requires a model to be trained on the specific data a document contains. As the new models keep being released, we have to keep updating our data.

1.3 Problem Statement

Since ROUGE [6] only considers N-gram lexical overlap, abstractive summaries conveying the same meaning but containing different vocabulary of words may get wrongly penalized. There may also be the case that there's high lexical overlap between candidate and reference but maybe conveying different meanings.

Table 1.1: Instances where ROUGE fails

index	**type**	**example**	**LO**	**CS**
1	**ref**	The quick brown fox jumped over a lazy dog	L	H
	cand	The fast wood-colored fox hopped over a lethargic dog		
2	**ref**	The weather is cold today	L	H
	cand	It is freezing today		
3	**ref**	The quick brown fox jumped over a lazy dog	H	L
	cand	The quick brown dog jumped over a lazy fox		

ref- reference, cand- candidate, LO- Lexical Overlap, CS- Contextual Similarity, L- Low, H- High

The above instances indicate that capturing only the content overlap does not assess factual and semantic correctness in the candidate summary. An ideal evaluation metric should look for qualitative measures like Grammatical correctness, Arrangement of sentences, Text Quality (Quality of language used and suitability with a set of users the application serves), Coherence (Conciseness and Exhaustiveness), or at least be a good proxy of them.

Based on this we raise a research question: "Does any evaluation metric exist which can act as a proxy to human evaluation?"

1.4 Key Contributions

Key contributions of this thesis are:

1. A Comparative study of correlation of human judgments with 40 automatic evaluation metrics. (Refer chapter 7).

2. Human judgments dataset for machine-generated abstractive summaries of Indian News data. (Refer chapter 4)

CHAPTER 2

Literature Survey

This chapter covers details about the Literature survey. We cover Transformers, Different models of extractive and Abstractive Summarization (Transformers and Non-Transformers based), Transformer based models to compute contextual embeddings, Traditional Word Embedding Models, Evaluation Metrics and Correlation methods.

2.1 Transformers

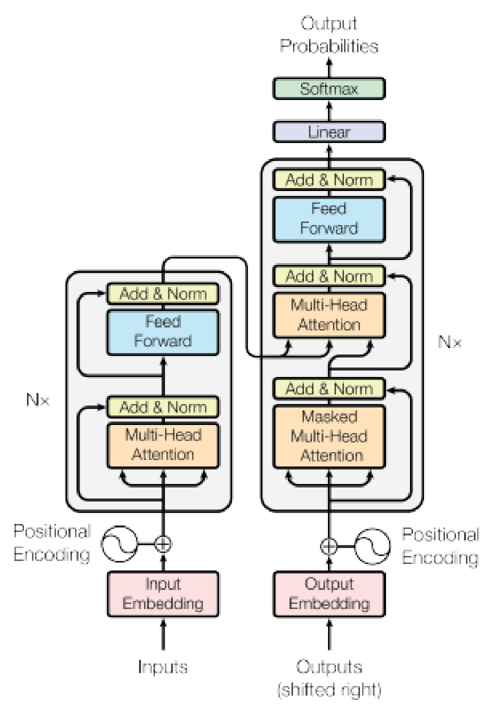

Figure 2.1: Transformers Architecture

Vaswani et al. (2017) [18] propose an attention mechanism-based encoder-decoder-based architecture called "transformers" for sequence processing that replaces standard recurrence-based architecture for the machine translation task. The encoder is a stack of n encoder blocks containing multi-head attention and a position-wise fully connected feed-forward network which can process the input in bidirectional way. The decoder is a stack of n decoder blocks containing masked multi-head (unidirectional) attention, multi-head attention (same as earlier) and feed-forward network. They reported BLEU scores of 28.4 and 41.0 for English to German and French, respectively.

2.2 Contextual Embedding models

2.2.1 Word Embedding Models

word2vec

word2vec [9] was one of the first methods to generate contextualized word embeddings trained using a word set of 1.6 billion words. The model was trained using a proxy task of predicting a middle word using context words or vice versa called CBOW and SkipGram respectively.

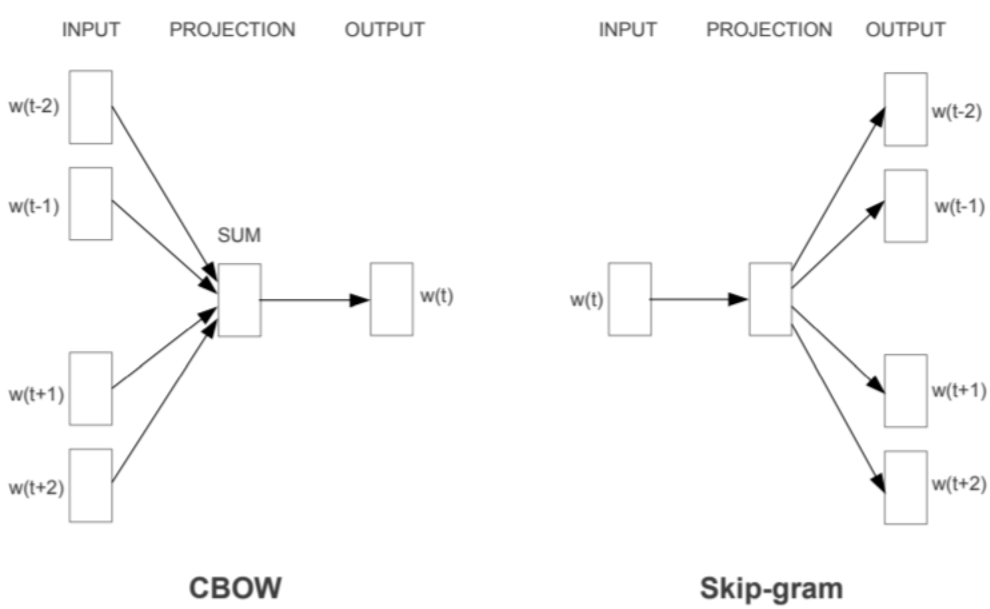

Figure 2.2: Word2vec Architecture

GloVe

GloVe [12] is an unsupervised algorithm for computing word vector representations. Unlike word2vec, which uses local context, it uses global context, i.e., co-occurrence of words to obtain word representations.

2.2.2 Transformer-based contextual embedding methods

PEGASUS

Zhang et al. (2020) [20] use standard encoder-decoder-based transformer architecture and propose a new pre-training objective called Gap Sentence Generation (GSG), which replaces important sentences with [MASK] tokens and lets the model predict them in a self-supervised fashion. This pre-training objective closely resembles the process of extractive summarization. There are mainly three strategies that authors are proposing to select the sentences, Random (selects m sentences randomly), Lead (selects first m sentences), and Principal (greedily selects m sentences by maximizing the Rouge-f1). Authors consider HugeNews and C4 corpus for pre-training. The authors also took human feedback on the generated summary and found it closely resembling human-generated summaries. Pegasus proves to be performing well for low resource summarization; it demonstrated state-of-the-art performance in as low as 1000 data points on datasets including CNN/Daily mail and reported Rouge-1 of 47.21. Refer Fig. 2.3 for architecture.

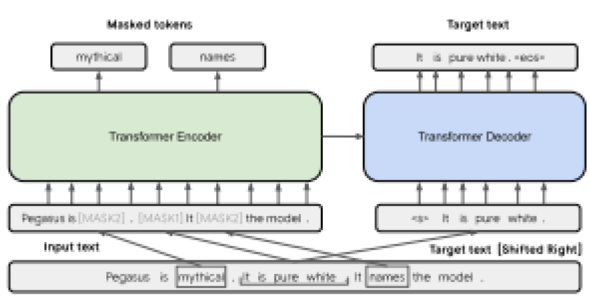

Figure 2.3: PEGASUS Architecture

T5

By proposing a uniform framework that transforms all text-based language problems into a "text-to-text" format, Colin et al. (2020) [15] investigate transfer learning strategies for NLP. This architecture enables us to use the same model, objective, training approach, and decoding process for all tasks considered by the model, including machine translation, question answering, abstractive summarization, and text classification. The authors focus on transfer learning and use models that process input with an encoder before creating an output with a separate decoder. The authors use the C4 corpus for pre-training and create an objective that randomly samples and then removes 15% of tokens from the input sequence. We supply our input text prefixed with the task name, e.g., "summarise: your text" to conduct any task. Refer Fig. 2.4 for Architecture.

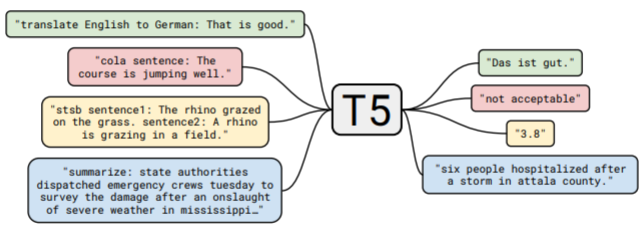

Figure 2.4: T5 Architecture

BART

Lewis et al. (2019) [5] introduce denoising autoencoders for sequence-to-sequence tasks based on simple transformer architecture. For pre-training, authors corrupt the input with arbitrary noise functions and make the model to predict the original text. Token masking, Text Infilling, token deletion, document rotation, and text permutation are the supported noise functions. Authors report a Rouge-1 score of 44.16 on the CNN/Daily Mail dataset. Refer Fig. 2.5 for Architecture.

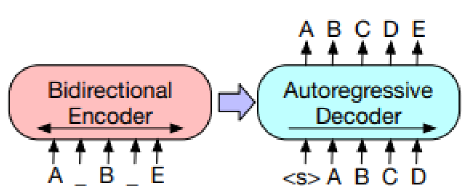

Figure 2.5: BART Architecture

BERT

BERT [1] stands for "Bidirectional Encoder Representations from Transformers, " built by stacking transformer encoders. As its name suggests, it is bidirectional and trained in 2 phases: pretraining and finetuning. It uses Masked language modeling(MLM) and Next sentence prediction (NSP) for pretraining. Task-specific training is possible by stacking layers on top of BERT architecture. It is available in two variants BERT-BASE and BERT-LARGE with 12 and 24 layers respectively.

RoBERTa

RoBERTa [8] stands for “A Robustly Optimized BERT Pretraining Approach” which has same architecture as BERT[5], but authors explore various design choices and re-examine pre-training and training procedures used in BERT. In its pre-training, authors use dynamic-maskinig instead of BERT-like masking.

DeBERTa

DeBERTa [4] stands for "Decoding-enhanced BERT with Disentangled Attention," which improves upon BERT and RoBERTa. The key improvements authors present are disentangled attention and enhanced mask decoder. They also present a network with the stacking of 48 encoder layers.

2.3 Extractive Summarization Models

2.3.1 BERTSUM

Liu, Yang et al. (2019) [7] propose an architecture called BertSum that uses BERT[5] for extractive summarization. The author argues that BERT, with its pre-training on large corpus and robust architecture for learning complex features, can improve the performance of extractive summarization. In BERTSUM, sentences are separated by [CLS] tokens and interval segment embeddings are added to generate a representation for each sentence. After getting sentence representations, several summarization-specific layers are added and are trained as a binary classification problem (i.e., whether to include it in summary or not).Authors experimented with a simple classifier, inter-sentence transformer, and RNNs. On the CNN/Daily Mail dataset, it was discovered that the BertSum + inter-sentence transformer combination produces the best results. Refer Fig. 2.6 for architecture.

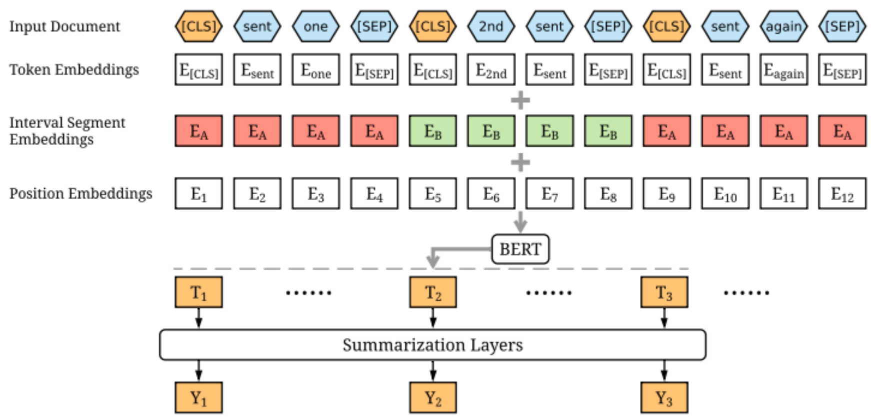

Figure 2.6: BERTSUM Architecture

2.3.2 BERT-KMEANS

Derek Miller et al. (2019) [10] propose an unsupervised technique for extractive summarization where he generates representation for sentences using BERT[5] and applies the KMeans algorithm on sentence representations. For generating summaries, it extracts sentences nearest from cluster centroids. The idea of clustering of sentences can be thought of as covering all the focus points of the source document(s). Refer Fig. 2.7 for architecture.

2.4 Abstractive Summarization Models

Above mentioned transformer-based models like PEGASUS, and BART can also be used for abstractive summarization, posing the problem as a seq-to-seq problem.

2.4.1 GET-TO-THE-POINT

See et al. (2017) [17] propose using a pointer-generator network for accurate reproduction of text and coverage mechanism for keeping track of what has been already generated. The approach suggested in this paper uses the standard LSTM based seq-to-seq model. Authors report the Rouge-1 of 39.53 on CNN/Daily mail dataset. Refer Fig. 2.7 for architecture.

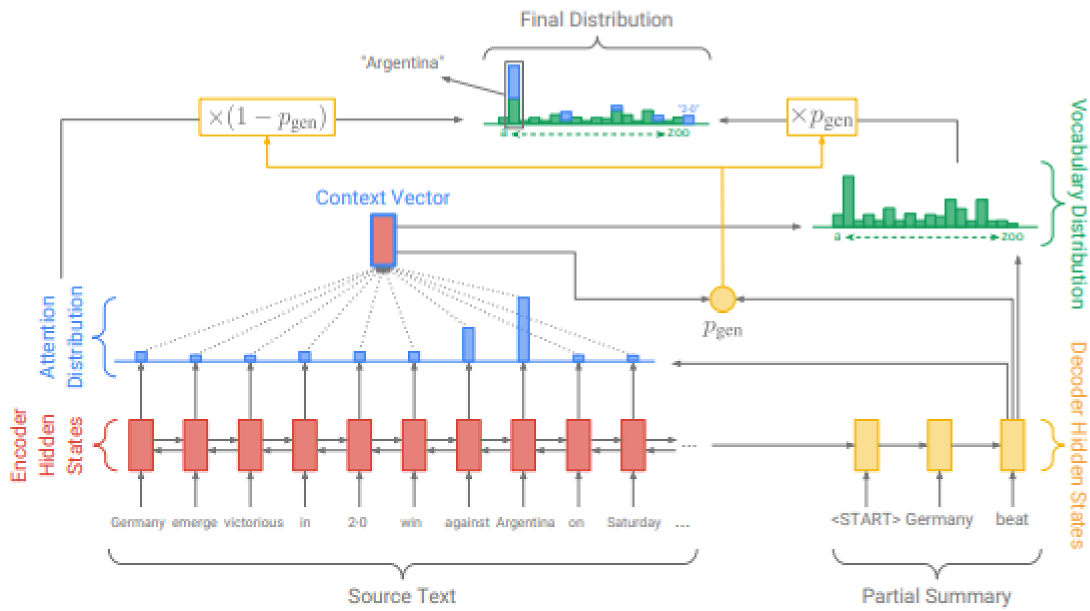

Figure 2.7: GET-TO-THE-POINT Architecture

Table 2.1: Comparison of Summarization Algorithms in Terms Of ROUGE

Model	**Type**	**ROUGE-1**	**ROUGE-2**
BERTSUM	Extractive	43.25	20.24
GET-TO-THE-POINT	Abstractive	39.53	17.28
PEGASUS	Abstractive	47.21	21.47
BART	Abstractive	44.16	21.28

2.5 Evaluation Metrics

2.5.1 Lexical Overlap

ROUGE

Lin et al. (2004) [6] define Rouge as Recall-Oriented Understudy for Gisting Evaluation. It measures N-gram overlap between system-generated summary and reference summary. It is computed in 3 variants: precision, recall, and f1.

$$\text{Rouge}_{\text{ precision}} = \frac{x_{\text{overlap}}}{x_{\text{system}}}$$

$$\text{Rouge}_{\text{ recall}} = \frac{x_{\text{overlap}}}{x_{\text{reference}}}$$

$$\text{Rouge}_{\ F1} = 2 * \frac{\text{Rouge}_{\text{ precision}} * \text{Rouge}_{\text{ recall}}}{\text{Rouge}_{\text{ precision}} + \text{Rouge}_{\text{ recall}}}$$

where,

$$x_{\text{overlap}} = \text{number of overlapping ngrams}$$
$$x_{\text{system}} = \text{number of ngrams in generated summary}$$
$$x_{\text{reference}} = \text{number of ngrams in refe summary}$$

2.5.2 Contextual Similarity

For contextual similarity-based metrics, we project candidate and reference summaries on embedded space and obtain vectors for both. Then we compute the similarity between both the vectors. Multiple ways to project both texts on embedding space include methods explained in section 2.2.

BERTSCORE

Zhang et al. (2019) [21] propose a metric called BERTSCORE, which computes a similarity score for each token in the candidate sentence with each token in the reference sentence. It measures contextual similarity between two texts which do not necessarily have word or n-gram overlap. It has been used majorly in machine translation problems. Tokens can be optionally weighted by their respective IDF scores.

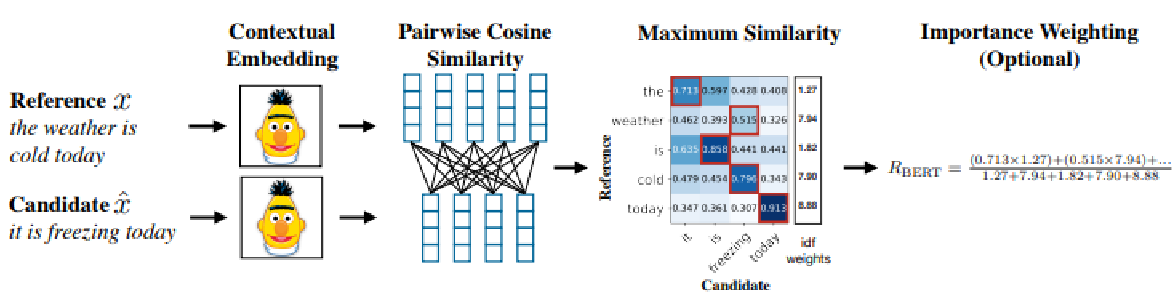

Figure 2.8: BERTScore computation

$$P_{BERT} = \frac{1}{|Y|} * \sum_{x_i \in x} \max_{y_j \in y} x_i y_j$$

$$R_{BERT} = \frac{1}{|X|} * \sum_{x_i \in x} \max_{y_j \in y} x_i y_j$$

$$F_{BERT} = 2 * \frac{R_{BERT} * P_{BERT}}{R_{BERT} + P_{BERT}}$$

Here, X is the reference summary, and Y is system generated summary.

CHAPTER 3

Background

3.1 Correlation Methods

3.1.1 Pearson Correlation

Pearson correlation is a measure of linear correlation between two variables. It's the ratio of two variables' covariances to the product of their standard deviations; it's effectively a normalised measurement of covariance. The result always ranges between -1 and 1.

$$r = \frac{\sum \left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum \left(x_i - \bar{x}\right)^2 \sum \left(y_i - \bar{y}\right)^2}}$$

r = correlation coefficient, x_i = values of the x-variable in a sample, $\bar{x}$ = mean of the values of the x-variable, y_i = values of the y-variable in a sample, $\bar{y}$ = mean of the values of the y-variable.

3.1.2 Spearman Correlation

The Spearman correlation between two variables is calculated as the Pearson correlation between the rank values of those variables. Unlike Pearson's correlation method, which assesses linear relationships, Spearman's correlation assesses monotonic relationships.

$$\rho = 1 - \frac{\sigma \sum d_i^2}{n\left(n^2 - 1\right)}$$

ρ = Spearman's rank correlation coefficient, d_i = difference between the two ranks of each - observation, $\boldsymbol{n}$ = number of observations.

3.2 Vector similarity measures

3.2.1 Cosine Similarity

It determines whether two vectors are pointing in the same general direction by measuring the cosine of the angle between them. It is determined by the angle of the vectors rather than their magnitudes.

$$\text{cosine similarity } = S_z(x, y) := \cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}$$

$$\text{x} = \text{vector-1}, \text{y} = \text{vector-2}, S_z(x, y) = \text{cosine similarity between x and y}$$

3.2.2 Earth Mover's Distance

In this setup, we consider vectors as point clouds or piles. The Earth Mover's Distance is the lowest cost of changing one pile into the other, where the cost is considered to equal the amount of dirt transported times the distance travelled. The weight appears to flow from one distribution to the next until they are identical, just like filling holes with dirt piles. Let A, B be two distributions such that: $\forall x \in A$ and $\forall y \in B$ have probability mass functions $p_x = \frac{1}{|A|}$ and $p_y = \frac{1}{|B|}$

Let $d_{x,y} =$ Euclidian distance between x and y
Let $z_{x,y} =$ amount of dirt to move from $x \in A$ and $y \in B$, with cost d(x, y)
Let $H(A, B) =$ all feasible flows between A, B

Let WORK(F,A,B) be the amount of dirt transported between A, B using one of the Feasible flow F.

$$WORK(F, A, B) = \sum_{x=1}^{m} \sum_{y=1}^{n} z_{x,y} d_{x,y}$$

$$EMD(A, B) = \frac{min_{F=z_{x,y} \in H(A,B)} WORK(F, A, B)}{min(\sum_{x \in A} p_x, \sum_{y \in B} p_y)}$$

CHAPTER 4
Dataset

To compare the correlation of different evaluation metrics with human judgments, a dataset that contains human judgments for machine-generated summaries is needed. Due to the unavailability of such a dataset, we opted to create one. The NEWS SUMMARY [19] dataset, which contains news articles, news titles, and their human-written summary scraped from the News aggregator platform, InShorts[1] ; was chosen as a base dataset.

1. Only the subset of the dataset was retained, having tuples with total tokens in the source article between 200-500.

2. Abstractive summaries of source articles were generated using BART [5] finetuned on CNN/Daily Mail dataset [14].

At this point, the dataset had 1001 tuples containing the following fields: Source Article, Source Title, Human Written Summary , Machine-generated Summary (BART). For collecting the Human judgments (reviews) for Machine-generated summaries, Human annotators(reviewers) were presented with news titles, human-written summaries(to be considered a gold standard), and their respective machine-generated summaries. They were told to rate machine-generated summaries on some "Qualitative Measures" and rate them overall between 1-5. Since reviews can be subjective, two sets of reviews were collected. A total of 30 people took part in the annotation process, out of which nine people contributed in set-1 and 21 in set-2 (Refer Appendix A). Reviews were collected using an in-house tool, which we eventually open-sourced (Refer Appendix B.).

[1] https://www.inshorts.com/

4.1 Qualitative Measures

1. **Grammatical Correctness (GC):** A summary should be grammatically correct, i.e., there should not be any spelling mistakes, sentence formation should be correct, and punctuations should be appropriately used.

2. **Arrangement of sentences/Flow of information (AS):** A summary is essentially a 3-5 lines story. The arrangement of sentences should be such that the story makes sense.

3. **Text Quality (TQ):** A summary should contain a language that suits its target audience. In this case, it is news. News is should be neutral and should not contain aggressive language. News should be direct; it should not resort to trolling or sarcasm.

4. **Conciseness (CS):** A summary should be focused on a single topic, i.e., the topic presented in the title. It should not stumble between topics.

5. **Exhaustiveness (EX):** A summary should be exhaustive of all the needed details. Given the topic it covers, it should satisfy the reader's information need.

6. **Overall Rating (OR):** Based on the above-mentioned Qualitative measures, a reviewer was asked to rate the given summary.

The reason behind asking overall score was that we wanted to check whether asking for only the overall score is sufficient.

4.2 Annotation Results

Figure 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 represent the histograms of scores received for above mentioned qualitative points. It can be observed that there's minimal disagreement between the reviws of both sets.

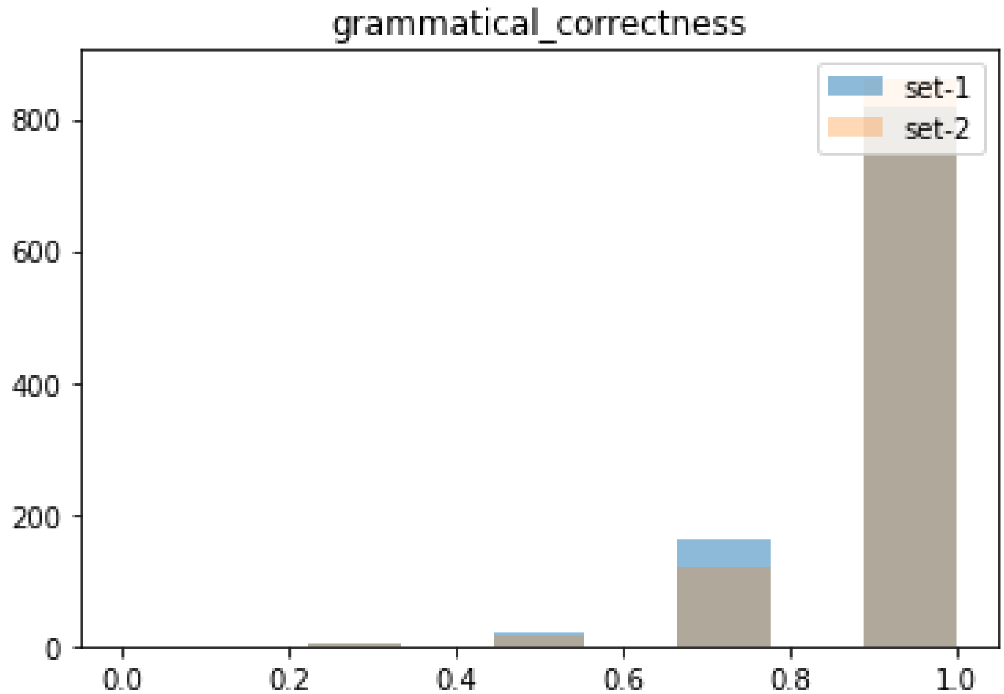

Figure 4.1: Histograms of "Grammatical Correctness" scores in both sets of reviews

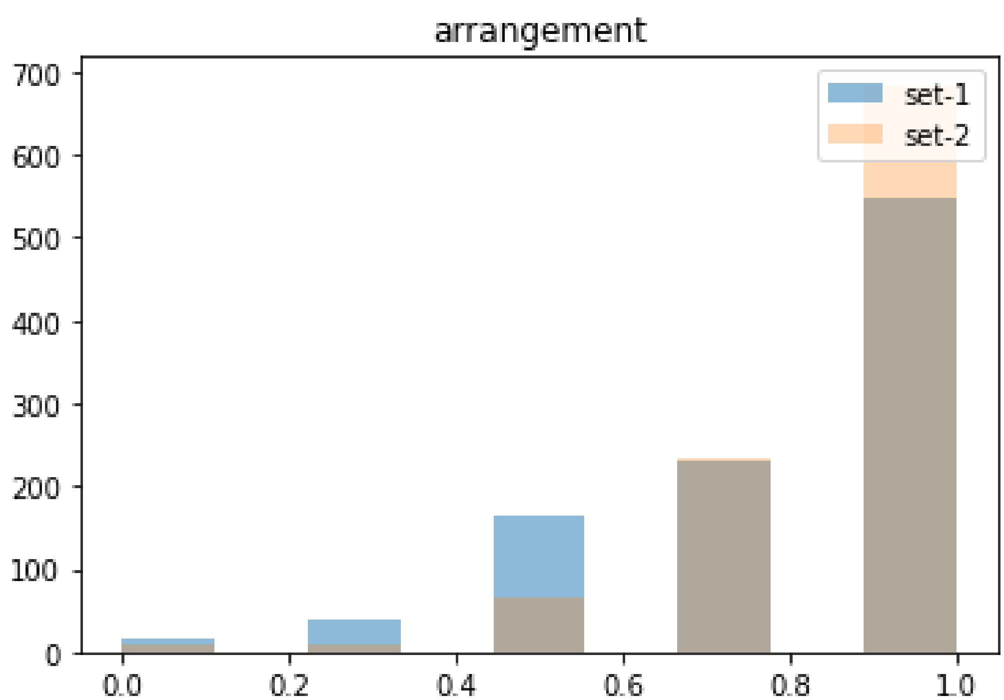

Figure 4.2: Histograms of "Arrangement of sentences/Flow of information" scores in both sets of reviews

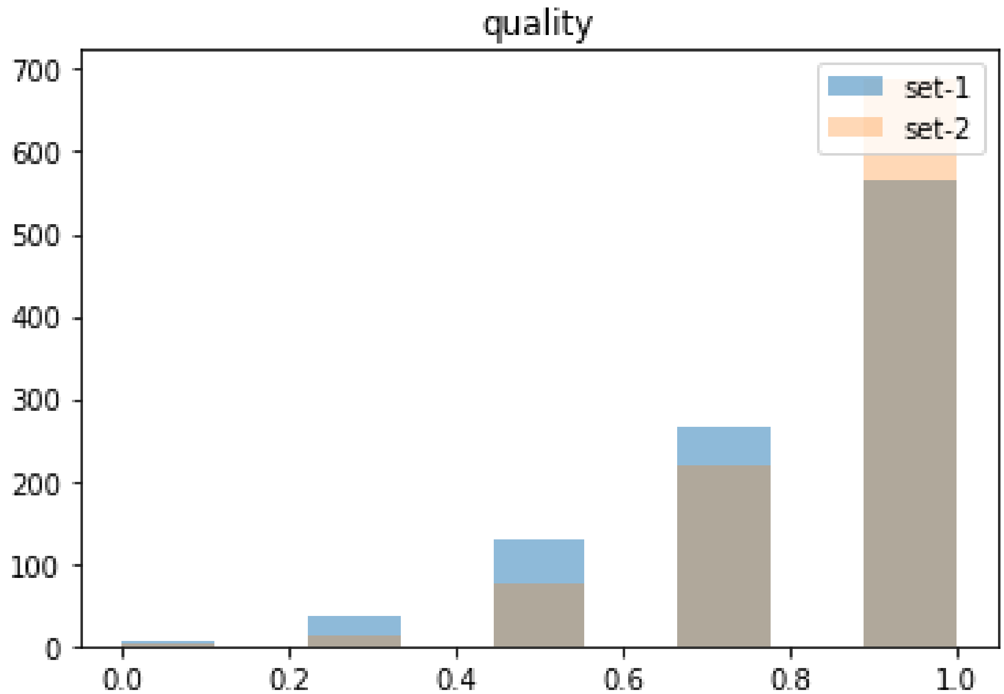

Figure 4.3: Histograms of "Text Quality" scores in both sets of reviews

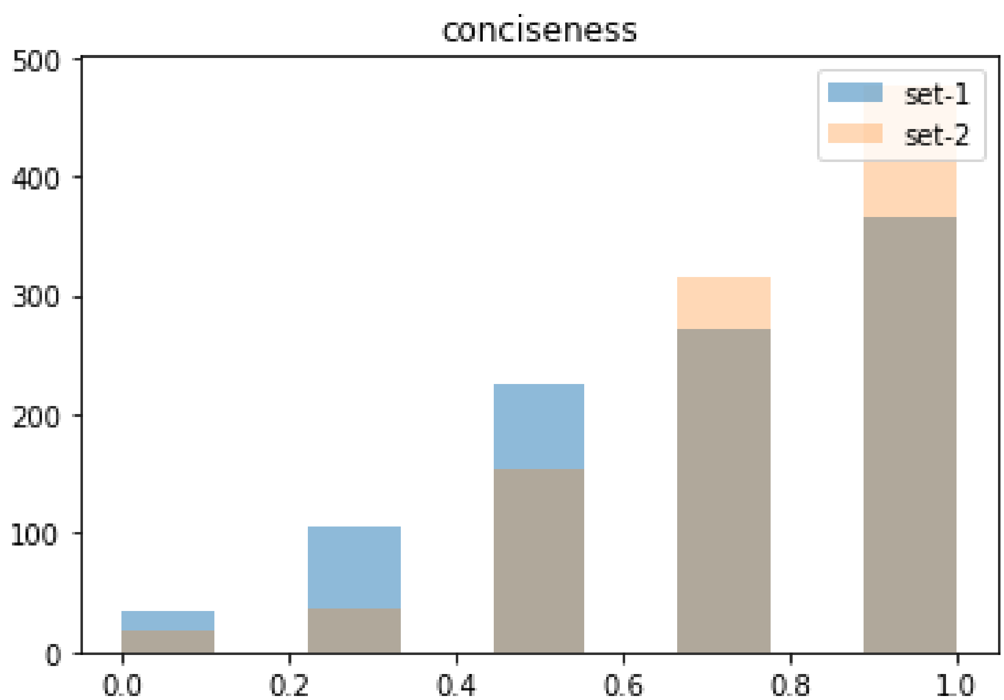

Figure 4.4: Histograms of "Conciseness" scores in both sets of reviews

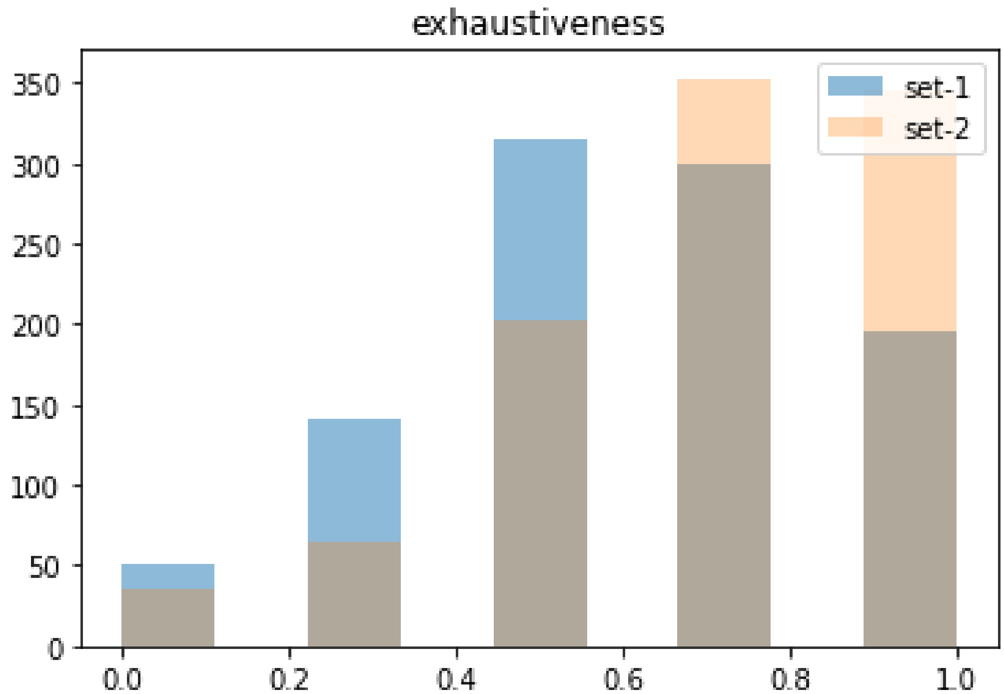

Figure 4.5: Histograms of "Exhaustiveness" scores in both sets of reviews

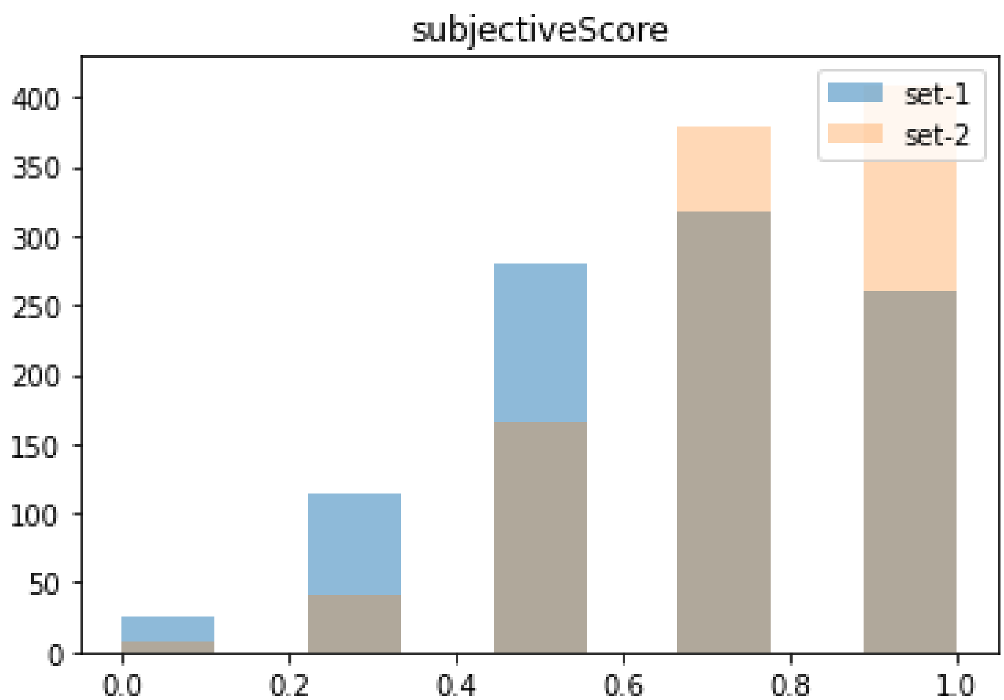

Figure 4.6: Histograms of "Overall Rating" scores in both sets of reviews

CHAPTER 5

Evaluation Metrics

Sections 5.2.1, 5.2.2, 5.2.3 use different variants of Transformer architecture. The models and model-weights were loaded from Transformers[1] library from Huggingface[2].

5.1 Lexical Overlap

5.1.1 ROUGE

Table 5.1: Lexical Overlap based metrics

ALIAS	**MODEL**
R1	ROUGE1
R2	ROUGE2
RL	ROUGEL

Lexical overlap-based metrics like ROUGE-1 and ROUGE-2 respectively capture 1-gram similarity, and 2-gram similarity between source and reference summaries, while ROUGE-L captures the longest common subsequence. Refer Table 5.1

[1] https://pypi.org/project/transformers/
[2] https://huggingface.co/

5.2 Contextual Similarity

5.2.1 Sentence Transformers

Table 5.2: Sentence Transformers based metrics

ALIAS	MODEL
CS1	sentence-transformers/ sentence-t5-xl
CS2	sentence-transformers/ sentence-t5-large
CS3	sentence-transformers/ multi-qa-MiniLM-L6-cos-v1
CS4	sentence-transformers/ distiluse-base-multilingual-cased-v1
CS5	sentence-transformers/ paraphrase-MiniLM-L6-v2

These siamese networks [14] compute document embeddings by applying average pooling over token embeddings and compute the cosine similarity between both vectors. Refer Table 5.2.

5.2.2 BERT-like Architectures

Table 5.3: BERT-like architecture based metrics

ALIAS	MODEL
CS6	bert-base-uncased
CS7	roberta-base

BERT-like architectures have special tokens like [CLS] and [SEP]. The [SEP] represents the end of the sequence, while [CLS] stores the cumulative representation of token embeddings until the [SEP] is encountered. Therefore, we can pre-pend [CLS] token in both candidate and reference summary to compute the sentence embeddings and then compute the cosine similarity. Refer Table 5.3.

5.2.3 BERTSCORE models

Table 5.4: BERTSCORE based metrics

ALIAS	MODEL
BS00	bert-base-uncased
BS01	bert-large-uncased
BS02	bert-base-cased-finetuned-mrpc
BS03	roberta-base
BS04	roberta-large
BS05	roberta-large-mnli
BS06	facebook/bart-base
BS07	facebook/bart-large
BS08	facebook/bart-large-cnn
BS09	facebook/bart-large-mnli
BS10	facebook/bart-large-xsum
BS11	t5-small
BS12	t5-base
BS13	t5-large
BS14	microsoft/deberta-base
BS15	microsoft/deberta-base-mnli
BS16	microsoft/deberta-large
BS17	microsoft/deberta-large-mnli
BS18	microsoft/deberta-xlarge
BS19	microsoft/deberta-xlarge-mnli
BS20	google/pegasus-xsum

Instead of representing the candidate and reference summary as a single vector and then computing the cosine similarity, this method computes pair-wise cosine similarity between the tokens of both texts. Here, Transformer-based models like BERT, RoBERTa, DeBERTa, PEGASUS and T5 are used.Refer Table 5.4.

5.2.4 Word Embedding Models + Cosine Similarity

Table 5.5: Word Embedding + Cosine Similarity based metrics

ALIAS	MODEL
CS8	spacy en_core_web_sm
CS9	spacy en_core_web_md
CS10	spacy en_core_web_lg
CS11	word2vec
CS12	glove_twitter_25

To obtain vectors for candidate and reference summaries, this method computes the average of word embeddings. Then, the cosine similarity is computed between the vectors. Here CS8, CS9, and CS10 are from SpaCy[3] Library. Refer Table 5.5.

5.2.5 Word Embedding Models + Earth Mover's distance

Table 5.6: Word embeddings + Earth Mover's distance-based metrics

ALIAS	MODEL
WMD00	en_core_web_md
WMD01	en_core_web_lg
WMD02	word2vec-google-news-300
WMD03	glove-twitter-25

These models represent the candidate and reference summary as a point cloud of word vectors and compute the earth mover's distance between them.

[3] https://pypi.org/project/spacy/

CHAPTER 6

Methology

The Idea: Though automatic metrics cannot replace the need for human judgments, one having a higher correlation with human judgments is a better proxy. Since ROUGE is the De'facto metric for summarization, we look for a metric that correlates better with human judgments, i.e., is a better proxy than ROUGE. The proposed approach is:

1. Try different approaches to form a "Human score", a representation of human review for a summary as a single score.

2. Compute the correlation of "Human score" with various evaluation metrics to know which one is the best proxy.

6.1 Human Score

Table 6.1: Human Scores

Human Score	**Description**
HS_1	Equal weights to all "Qualitative Measures"
HS_2	Unequal weights to all "Qualitative Measures", the least weight to "Grammatical Correctness"
HS_3	Overall Score

To represent human ratings as a single score, there are two options: computing the weighted average of the before-mentioned qualitative measures or using the overall score directly. Since two sets of reviews were collected, average of both sets is considered as a final human rating.

$$HS = \omega_1 * GC + \omega_2 * AS + \omega_3 * TQ + \omega_4 * CS + \omega_S * EX$$

Here, HS stands for Human Score. We computed two variants of the weighted human score, HS_1 and HS_2. In HS_1, we give equal weights to every measure. we set $\omega_i = 0.20$. In HS_2, we make use of the observation that most of the human ratings given for grammatical correctness were 4 or 5, which means the summaries produced by the algorithm was grammatically correct. To compute HS_2, we computed the correlation matrix of GC, AS, TQ, CS, and EX and collapsed the matrix by summing up the rows. We computed applied softmax on it to make weights sum to one. In this way, grammatical correctness would get the least weight.

$$HS_2 = \text{Softmax}\left(A^\top I\right)$$

where, $A =$ correlation matrix of size 5×5 and $I =$ vector of 1's of size 5×1

Apart from above mentioned qualitative measures, we also asked our users to rate the given summary between 1 to 5, keeping in mind the measures. This was done to check whether taking ratings for an overall score would be sufficient as reviewing process involving reviews for each measure is time-consuming. We call it HS_3.

6.2 Correlation

Since HS_1, HS_2, and HS_3 represent human judgments, their correlation was measured with all the evaluation metrics discussed above to comprehend which metric is the best proxy for human reviews. The correlation methods used:

1. Pearson Correlation
2. Spearman Rank Correlation

CHAPTER 7
Results

Since ROUGE is a de-facto evaluation metric for summarization, it is considered the baseline. Since R1 performed the best in the ROUGE family, we plot the difference plots considering the R1 as a reference. Metrics having bars on the positive side in the difference plot correlate better with human judgments than R1.

7.1 Pearson Correlation

7.1.1 Lexical Overlap

Table 7.1: Pearson correlation of "Lexical Overlap" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
R1	**0.391125**	**0.403563**	**0.405817**
R2	0.302982	0.312113	0.31682
RL	0.306813	0.316603	0.316652

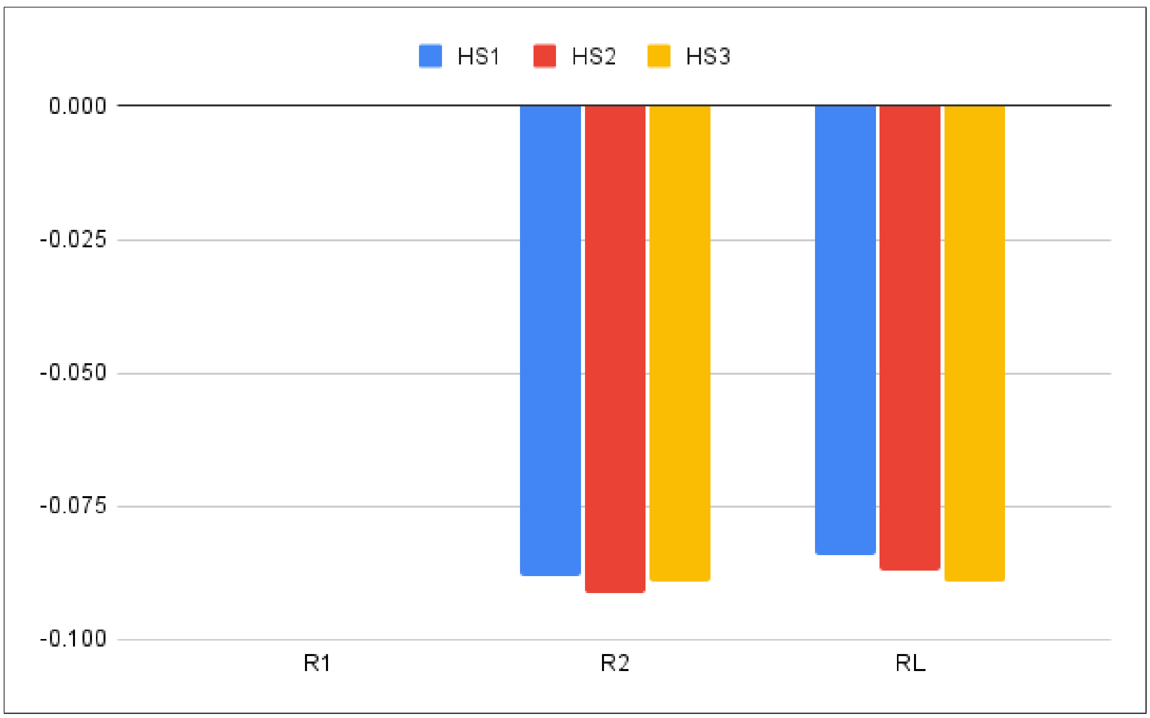

Figure 7.1: Difference Plot of Pearson correlation scores for "Lexical Overlap"

7.1.2 Sentence Transformers

Table 7.2: Pearson correlation of "Sentence Transformers" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
CS1	**0.474557**	0.489912	0.481815
CS2	0.473413	**0.490389**	**0.488624**
CS3	0.390508	0.402207	0.397415
CS4	0.424224	0.439674	0.426063
CS5	0.380785	0.39466	0.379361

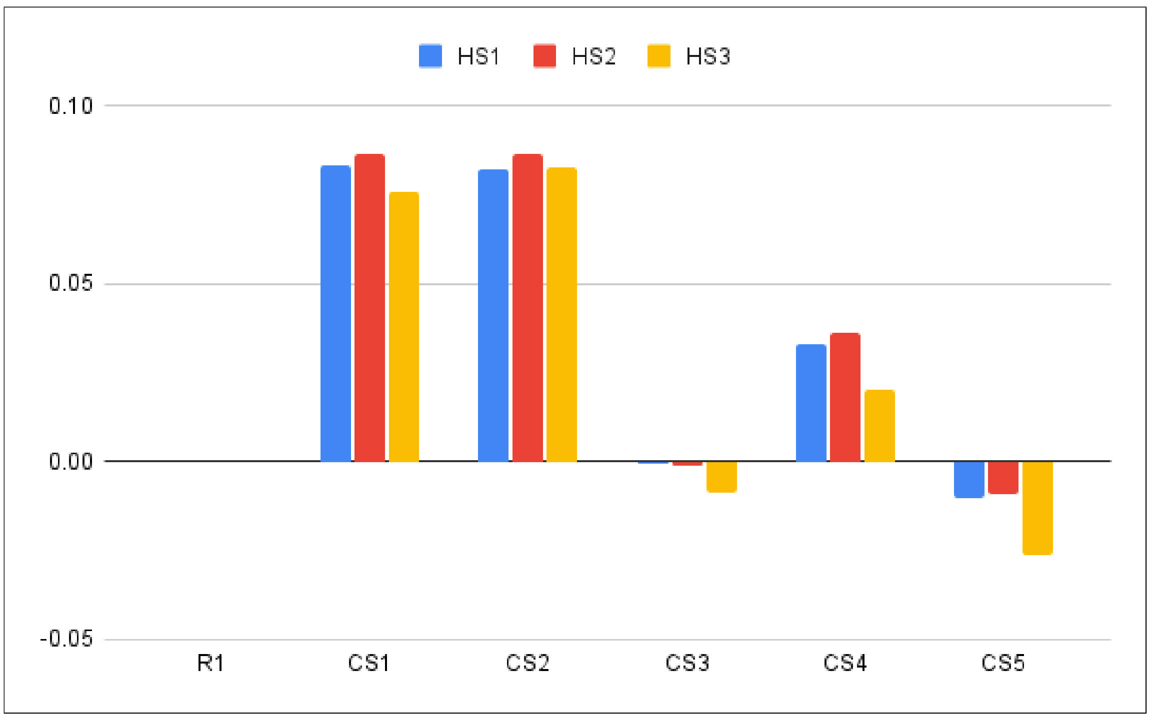

Figure 7.2: Difference Plot of Pearson correlation scores for "Sentence Transformers"

7.1.3 BERT-like Architectures

Table 7.3: Pearson correlation of "BERT-like Architectures" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
CS6	**0.310038**	**0.316681**	**0.333384**
CS7	0.307965	0.311064	0.327259

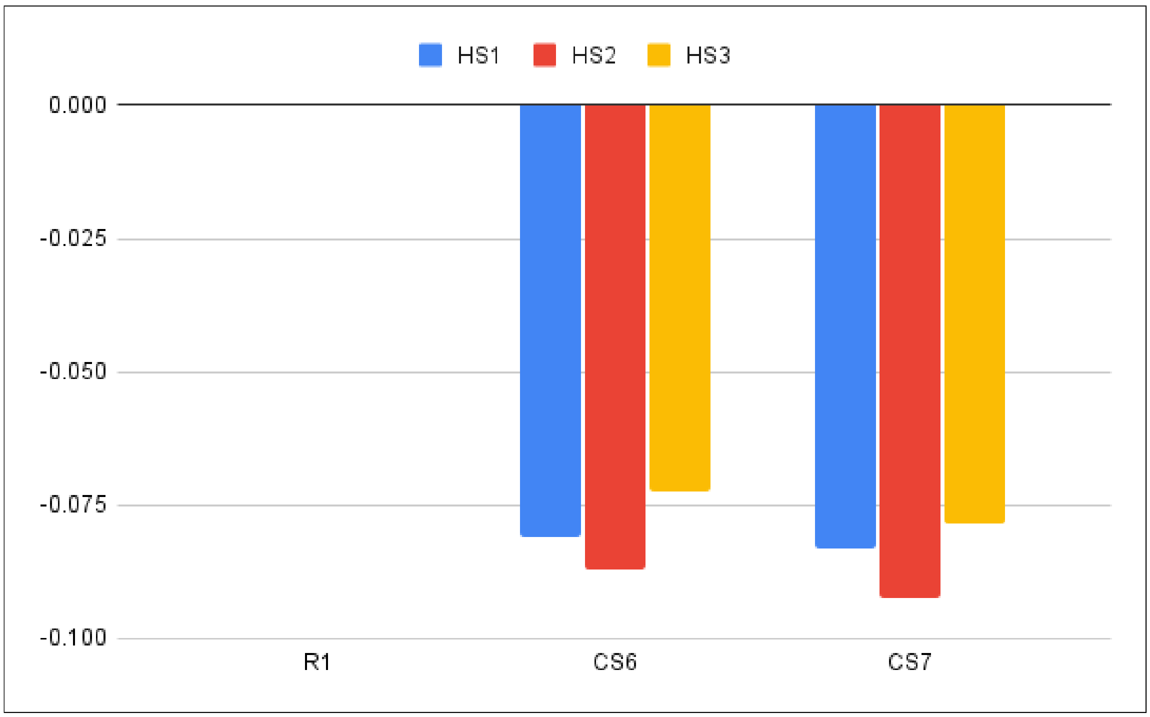

Figure 7.3: Difference Plot of Pearson correlation scores for "BERT-like Architectures"

7.1.4 BERTSCORE models

Table 7.4: Pearson correlation of "BERTSCORE models" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
BS00	0.420552	0.430709	0.431368
BS01	0.408182	0.418725	0.420935
BS02	0.383917	0.393258	0.390887
BS03	0.422546	0.433619	0.432300
BS04	0.417388	0.427617	0.428478
BS05	0.405949	0.418768	0.423377
BS06	0.421794	0.432247	0.432853
BS07	0.439262	0.450845	0.450916
BS08	0.426432	0.438161	0.440847
BS09	0.426478	0.438706	0.438667
BS10	0.446845	0.459058	0.463744
BS11	0.391053	0.402589	0.405289
BS12	0.411650	0.423299	0.424781
BS13	0.420807	0.433452	0.436602
BS14	0.414748	0.425428	0.428430
BS15	0.397459	0.409896	0.414107
BS16	0.433419	0.445226	0.445338
BS17	0.420299	0.433826	0.437837
BS18	0.430478	0.441647	0.440460
BS19	0.412649	0.426417	0.428735
BS20	**0.454003**	**0.467445**	**0.472490**

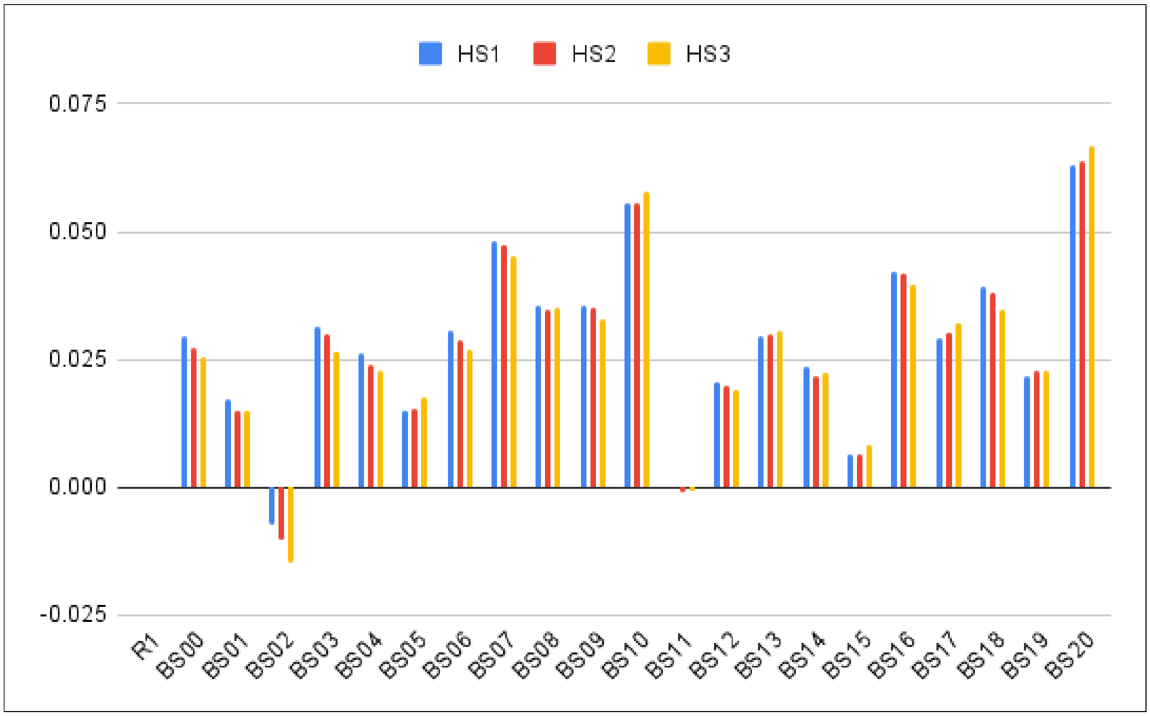

Figure 7.4: Difference Plot of Pearson correlation scores for "BERTSCORE models"

7.1.5 Word Embedding Models + Cosine Similarity

Table 7.5: Pearson correlation of "Word Embedding Models + Cosine Similarity" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
CS8	0.285575	0.290975	0.281239
CS9	0.331576	0.346874	0.339983
CS10	0.310512	0.325337	0.319883
CS11	**0.408396**	**0.423316**	**0.404882**
CS12	0.291956	0.305086	0.309118

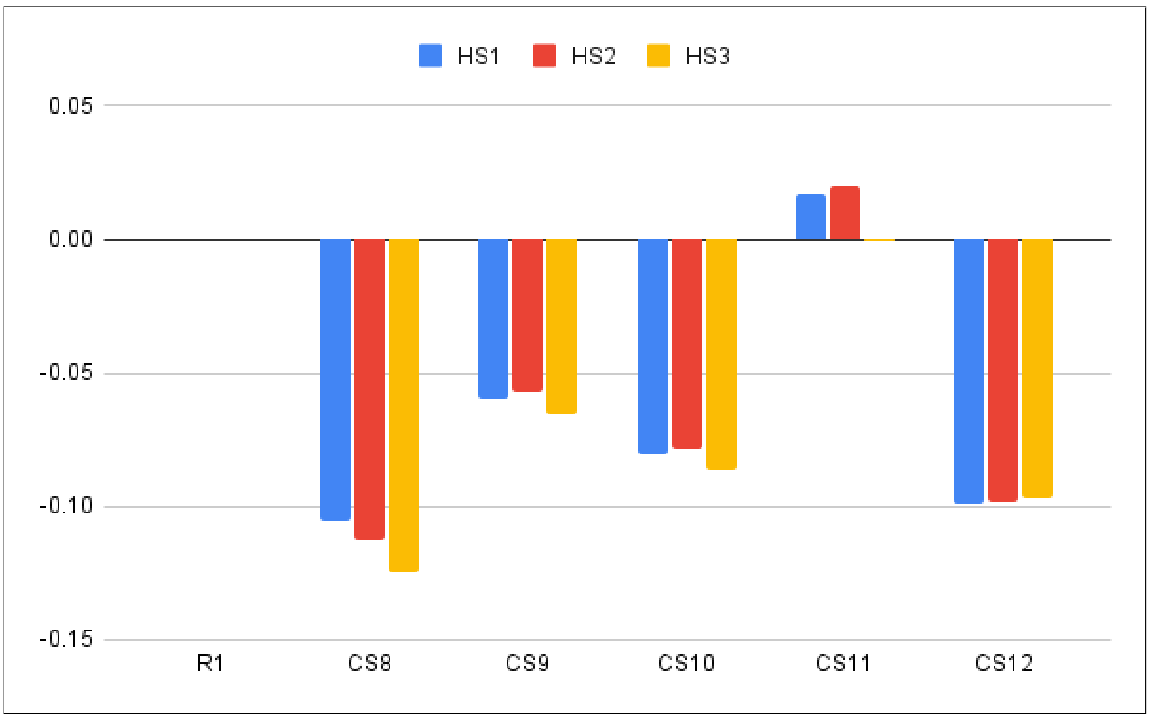

Figure 7.5: Difference Plot of Pearson correlation scores for "Word Embedding Models + Cosine Similarity"

7.1.6 Word embeddings + Earth Mover's distance

Table 7.6: Pearson correlation of "Word embeddings + Earth Mover's distance" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
WMD00	0.375220	0.385934	0.384831
WMD01	0.374999	0.385250	0.385092
WMD02	**0.381790**	**0.393033**	**0.386811**
WMD03	0.380668	0.391698	0.387522

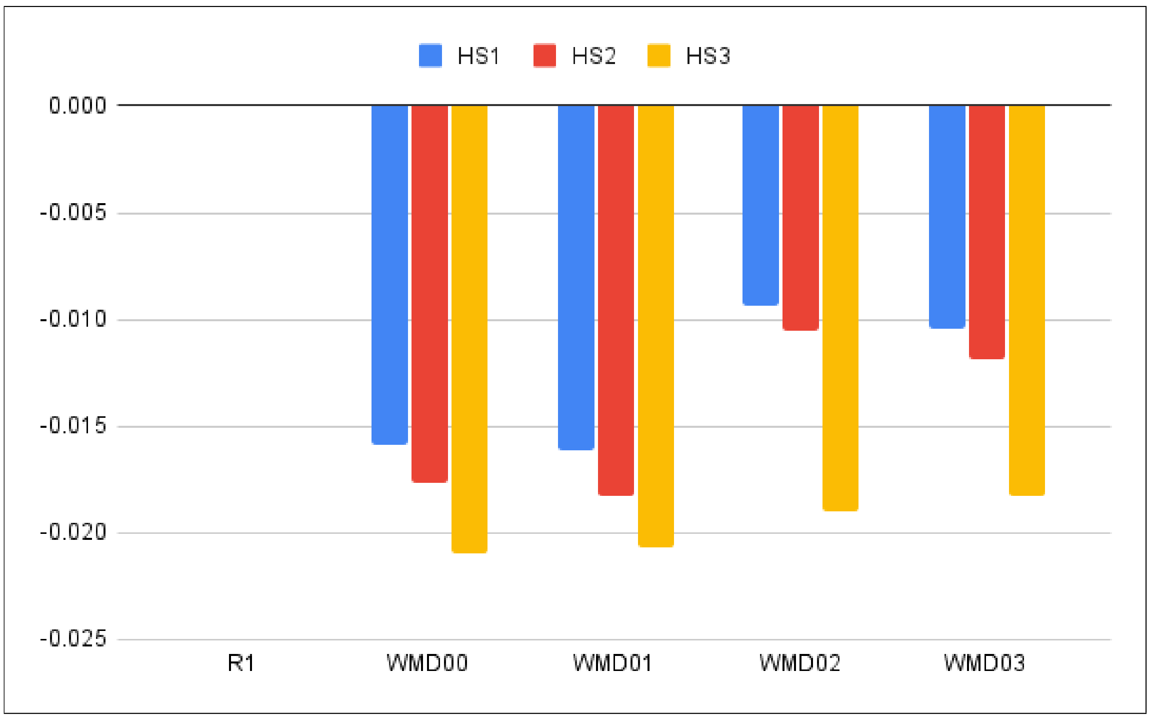

Figure 7.6: Difference Plot of Pearson correlation scores for "Word embeddings + Earth Mover's distance"

7.2 Spearman Rank Correlation

7.2.1 Lexical Overlap

Table 7.7: Spearman Rank correlation of "Lexical Overlap" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
R1	**0.379260**	**0.388557**	**0.400669**
R2	0.302302	0.311316	0.322189
RL	0.298070	0.306450	0.317079

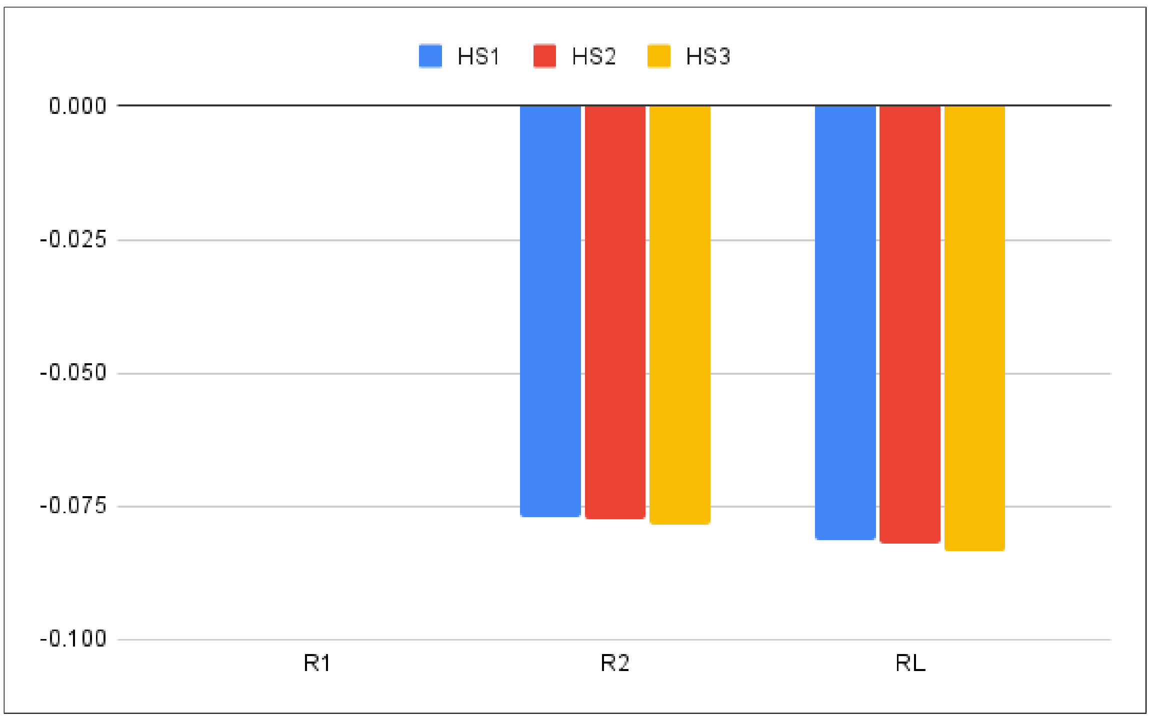

Figure 7.7: Difference Plot of Spearman Rank correlation scores for "Lexical Overlap"

7.2.2 Sentence Transformers

Table 7.8: Spearman Rank correlation of "Sentence Transformers" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
CS1	0.442113	0.454188	0.457508
CS2	**0.442481**	**0.455519**	**0.464508**
CS3	0.359298	0.367239	0.370118
CS4	0.393563	0.405466	0.403517
CS5	0.34961	0.359223	0.352005

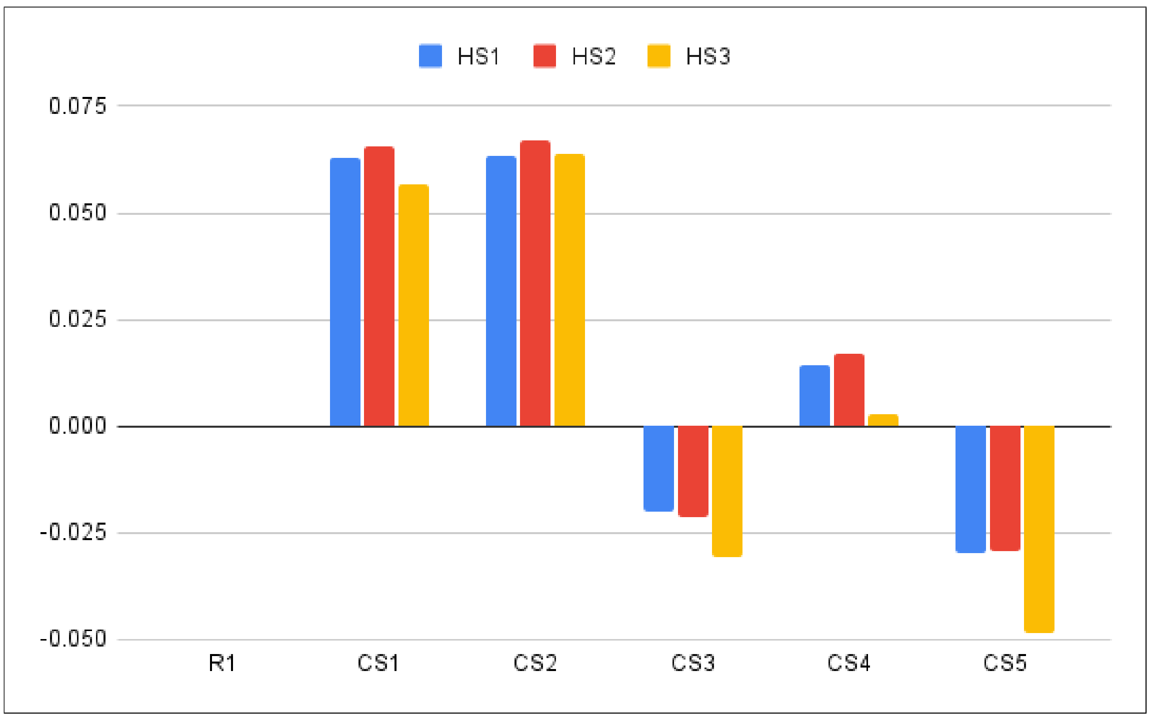

Figure 7.8: Difference Plot of Spearman Rank correlation scores for "Sentence Transformers"

7.2.3 BERT-like Architectures

Table 7.9: Spearman Rank correlation of "BERT-like Architectures" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
CS6	0.300683	0.305207	0.3219
CS7	**0.346404**	**0.344943**	**0.358525**

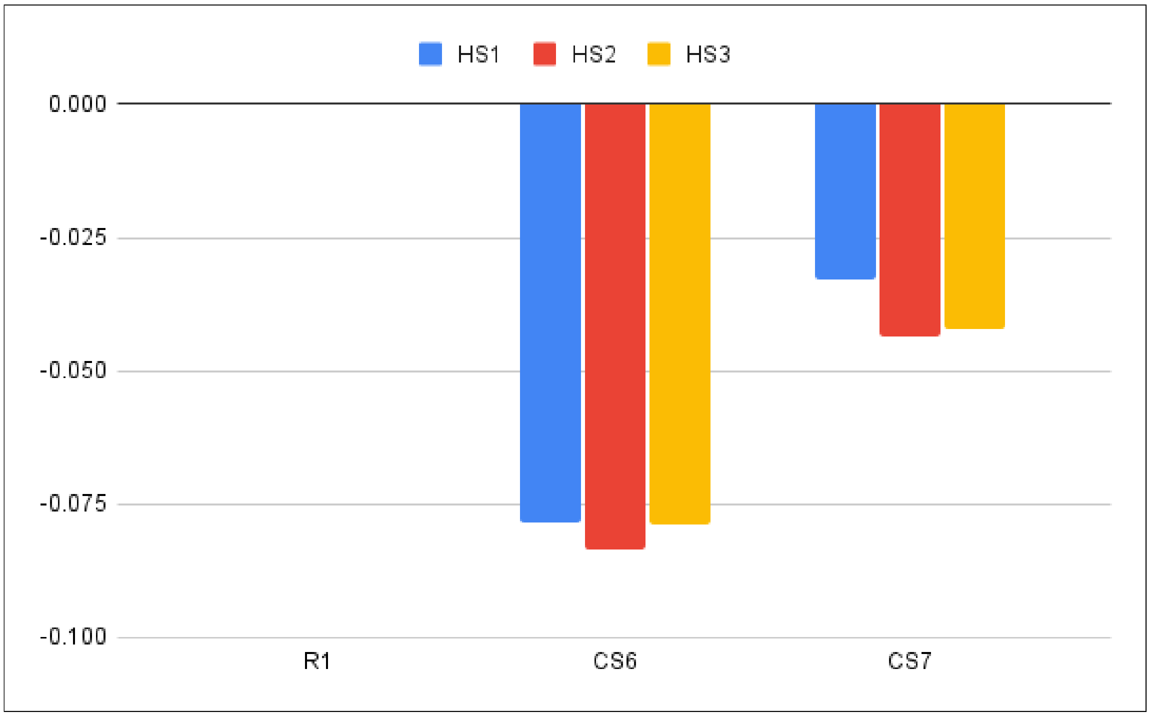

Figure 7.9: Difference Plot of Spearman Rank correlation scores for "BERT-like Architectures"

7.2.4 BERTSCORE models

Table 7.10: Spearman Rank correlation of "BERTSCORE models" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
BS00	0.420552	0.430709	0.431368
BS01	0.408182	0.418725	0.420935
BS02	0.383917	0.393258	0.390887
BS03	0.422546	0.433619	0.432300
BS04	0.417388	0.427617	0.428478
BS05	0.405949	0.418768	0.423377
BS06	0.421794	0.432247	0.432853
BS07	0.439262	0.450845	0.450916
BS08	0.426432	0.438161	0.440847
BS09	0.426478	0.438706	0.438667
BS10	0.446845	0.459058	0.463744
BS11	0.391053	0.402589	0.405289
BS12	0.411650	0.423299	0.424781
BS13	0.420807	0.433452	0.436602
BS14	0.414748	0.425428	0.428430
BS15	0.397459	0.409896	0.414107
BS16	0.433419	0.445226	0.445338
BS17	0.420299	0.433826	0.437837
BS18	0.430478	0.441647	0.44046
BS19	0.412649	0.426417	0.428735
BS20	**0.454003**	**0.467445**	**0.472490**

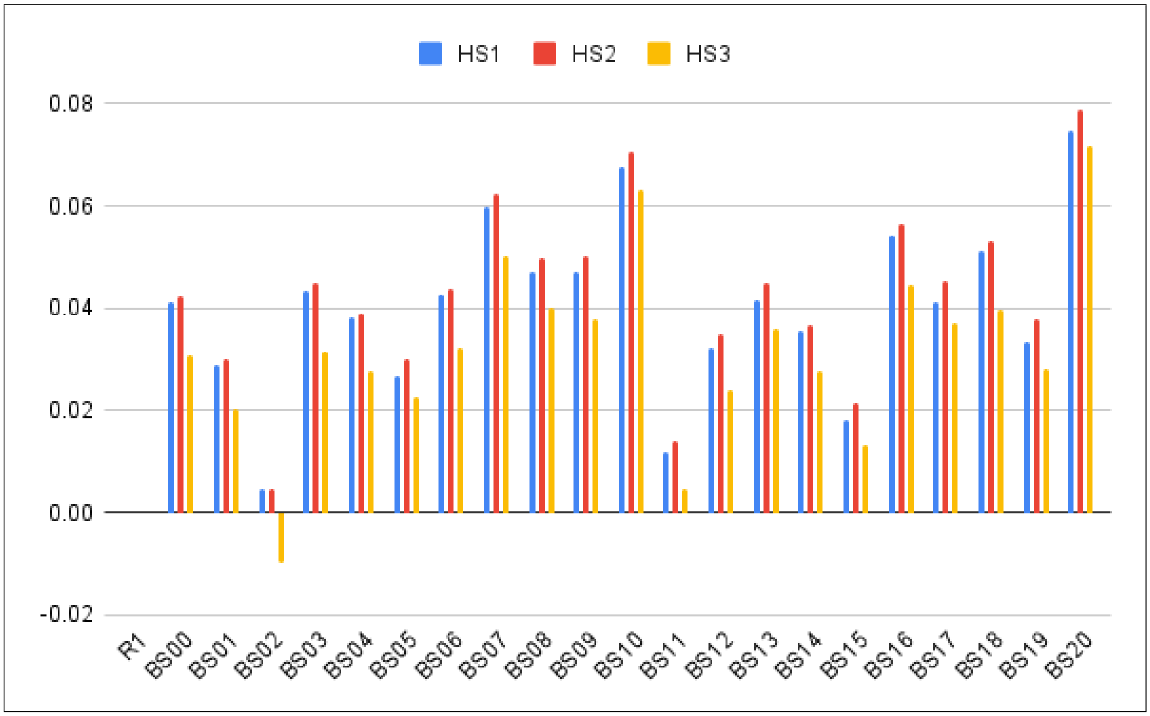

Figure 7.10: Difference Plot of Spearman Rank correlation scores for "BERTSCORE models"

7.2.5 Word Embedding Models + Cosine Similarity

Table 7.11: Spearman correlation of "Word Embedding Models + Cosine Similarity" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
CS8	0.285575	0.290975	0.281239
CS9	0.331576	0.346874	0.339983
CS10	0.310512	0.325337	0.319883
CS11	**0.408396**	**0.423316**	**0.404882**
CS12	0.291956	0.305086	0.309118

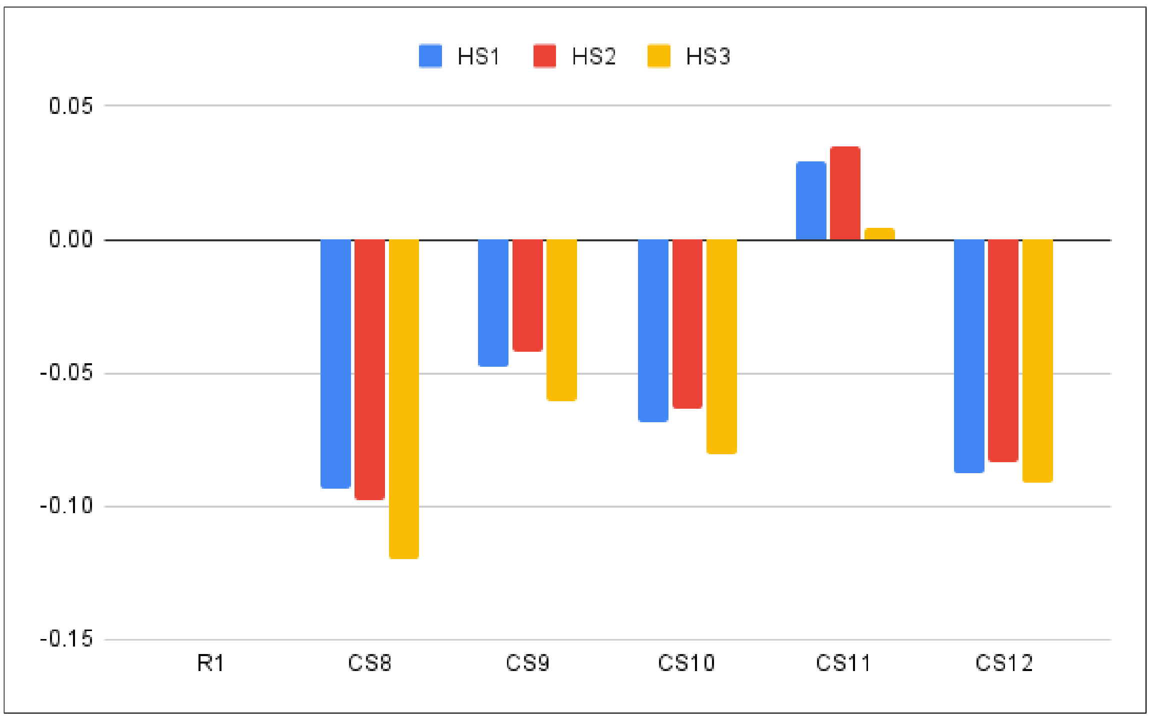

Figure 7.11: Difference Plot of Spearman Rank correlation scores for "Word Embedding Models + Cosine Similarity"

7.2.6 Word embeddings + Earth Mover's distance

Table 7.12: Spearman Rank correlation of "Word embeddings + Earth Mover's distance" based metrics with human judgments

Metric	HS_1	HS_2	HS_3
WMD00	0.37522	0.385934	0.384831
WMD01	0.374999	0.385250	0.385092
WMD02	**0.381790**	**0.393033**	0.386811
WMD03	0.380668	0.391698	**0.387522**

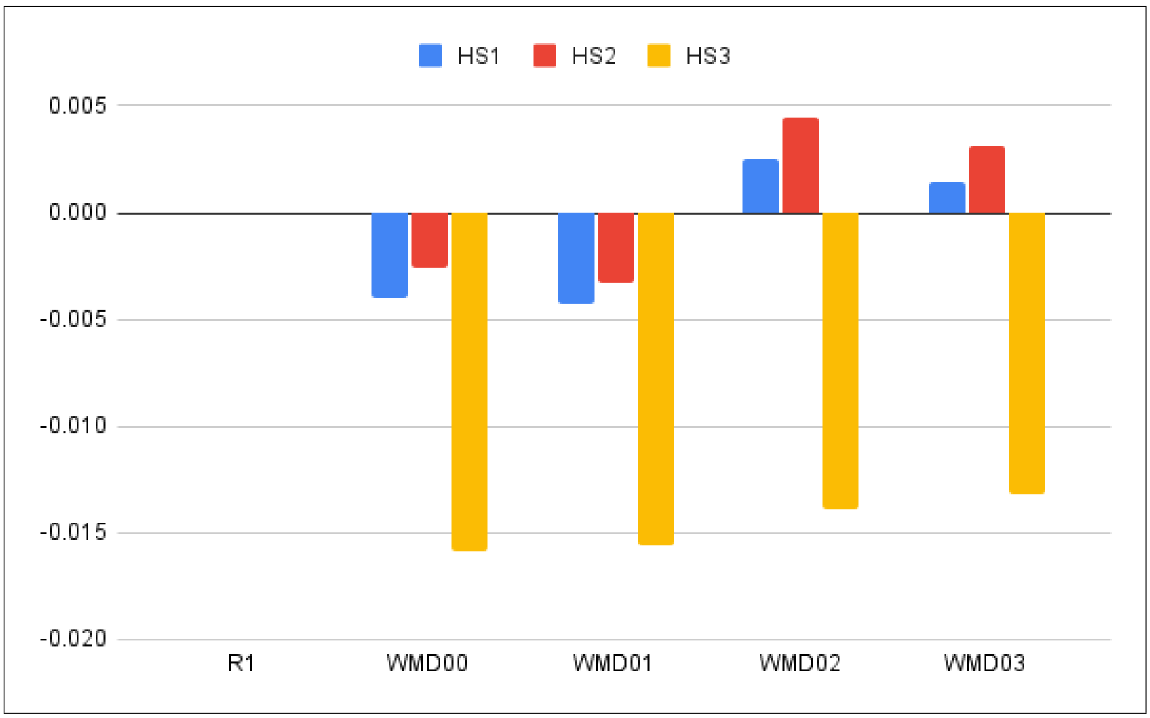

Figure 7.12: Difference Plot of Spearman Rank correlation scores for "Word embeddings + Earth Mover's distance"

7.3 Observations

Table 7.13: Best performing metrics

Correlation Method	HS_1	HS_2	HS_3
Pearson Correlation	CS1	CS2	CS2
Spearman Rank Correlation	CS2	CS2	CS2

As can be observed in Tables 7.1,7.7, and Figures 7.1,7.7, the best performing metric from the "Lexical Overlap" family, is R1, i.e., ROUGE-1. Thus, R1 is considered a baseline for comparison between all families of metrics.

Based on Table 7.5,7.11 and Figure 7.5,7.11, it can be observed that except for CS11 (word2vec), the method of averaging the word embedding to extract document embedding fails to capture the context; hence doesn't prove to be better proxy compared to ROUGE.

Tables 7.2,7.8, and Figures 7.2,7.8 indicate that these sentence transformers perform well except for CS3 (a multilingual model) and CS5. The best performing models from all 40 evaluation metric belong to this family, CS1, and CS2 (Table 7.13). CS1 is "sentence-transformers/sentence-t5-xl" and CS2 is "sentence-transformers/sentence-t5-large". Both are Sentence Transformer models and variants of T5 Architecture having 24 and 48 layers (in both encoder and decoder), respectively.

As seen in Tables 7.3,7.9, and Figures 7.3,7.9, the "[CLS]" token isn't able to capture the context of the document efficiently, and hence the methods that use it to represent the documents are not a good proxy.

Tables 7.6,7.12, and Figures 7.6,7.12 indicate that the idea of representing a document as a point cloud of word embeddings and computing the earth mover's distance doesn't prove to be a good proxy.

As observed in Tables 7.4,7.10, and 7.4,7.10, the idea of computing pairwise cosine similarity between tokens of reference and candidate summary proves to be a good proxy, except for BS02.

CHAPTER 8
Conclusions and Future Work

8.1 Conclusions

Based on the experimental results, we can conclude that some methods of capturing contextual similarity are a good proxy for human evaluation and even beat the de-facto metric ROUGE.

8.2 Future Work

The performance of contextual similarity methods largely depends on the architecture of the model, the data it is trained on, and the method of training (self-supervised objective in the case of transformer-based models). A central repository should keep track of the correlation of contextual similarity methods as the new model architectures, datasets, and training methods keep getting introduced. THe proposed experiment can be replicated on a larger scale, with a larger dataset and diverse reviewers for more robust results.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

[3] D. Gnatyshak, D. Garcia-Gasulla, S. Alvarez-Napagao, J. Arjona, and T. Venturini. Healthy twitter discussions? time will tell. *arXiv preprint arXiv:2203.11261*, 2022.

[4] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[5] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[6] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[7] Y. Liu. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*, 2019.

[8] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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

[10] D. Miller. Leveraging bert for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*, 2019.

[11] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[12] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[13] D. Radev, E. Hovy, and K. McKeown. Introduction to the special issue on summarization. *Computational linguistics*, 28(4):399–408, 2002.

[14] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[15] A. Roberts, C. Raffel, K. Lee, M. Matena, N. Shazeer, P. J. Liu, S. Narang, W. Li, and Y. Zhou. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019.

[16] S. Ruder. NLP-progress: Summarization. `http://nlpprogress.com/english/summarization.html`.

[17] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[19] K. VONTERU. NEWS SUMMARY. `https://www.kaggle.com/datasets/sunnysai12345/news-summary`.

[20] J. Zhang, Y. Zhao, M. Saleh, and P. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

[21] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Chapter A
Annotation Process

Table A.1: Reviewers who participated in Set-1

Reviewer	Qualification
Darshil Patel	MTech
Prahar Pandya	MTech
Dhyanil Mehta	MTech
Devansh Choudaha	MTech
Kishan Vaishnani	MTech
Tarang Ranpara	MTech
Shradha Makhija	MTech
EVV Haricharan	MTech
Sana Baid	MTech

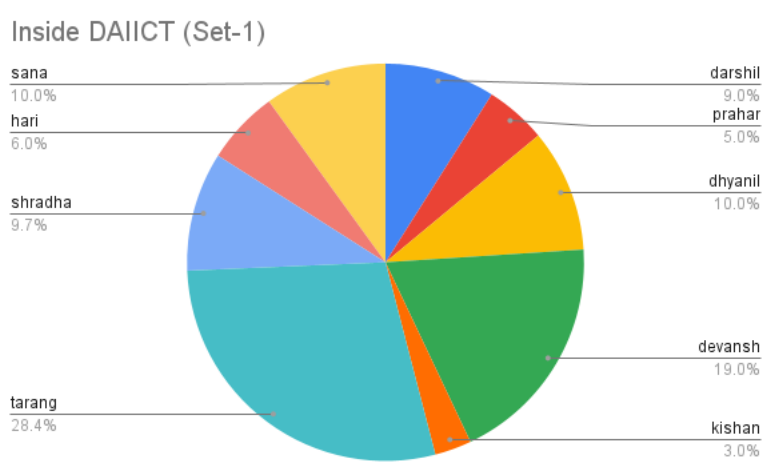

Figure A.1: Individual contribution of annotators in Set-1

Table A.2: Reviewers who participated in Set-2

Reviewer	**Qualification**
Kaushal Pandya	BTech
Dev Dave	BTech
khushali Ratanghayara	MTech
Harsh Bhatt	BTech
Ravi Dave	BTech
Mihir Kotecha	BTech
Raj Shah	MTech
Vikas Gajera	BArch
Jainil Soni	BTech
Kevin Jadia	MTech
Jugal Adesara	MBA
Jilsa Chandarana	BTech
Dev Parmar	BTech
Shivangi Gajjar	MTech
Viranya Shah	MS
Nupur Mehta	BTech
Dreamy Pujara	BTech
Rahul Vansh	MTech
Meet Shah	MTech
Krunal Ranpara	CA
Mrudang Vakharia	BTech

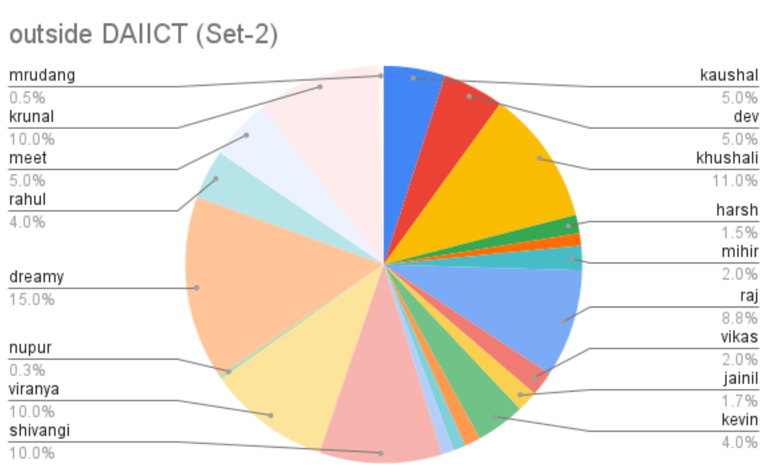

Figure A.2: Individual contribution of annotators in Set-2

Chapter B

Annotation Platform

Human reviews were collected from an in-house tool, built using Django, PostgreSQL hosted on two separate Linux servers. A tool is developed to be a de-facto annotation tool for any NLP task. It essentially works as a "work allocation system" where the admin can add/remove data samples(tasks), add/remove users, allocate/de-allocate tasks to users, and check the status of allocated work. The user can log in and complete assigned work across multiple sessions on the user's side. The platform can be tweaked to suit any NLP annotation task with minimal tweaking. Open-Source Repository: https://github.com/TarangRanpara/SummaryAnnotatorTool

B.1 User's Side

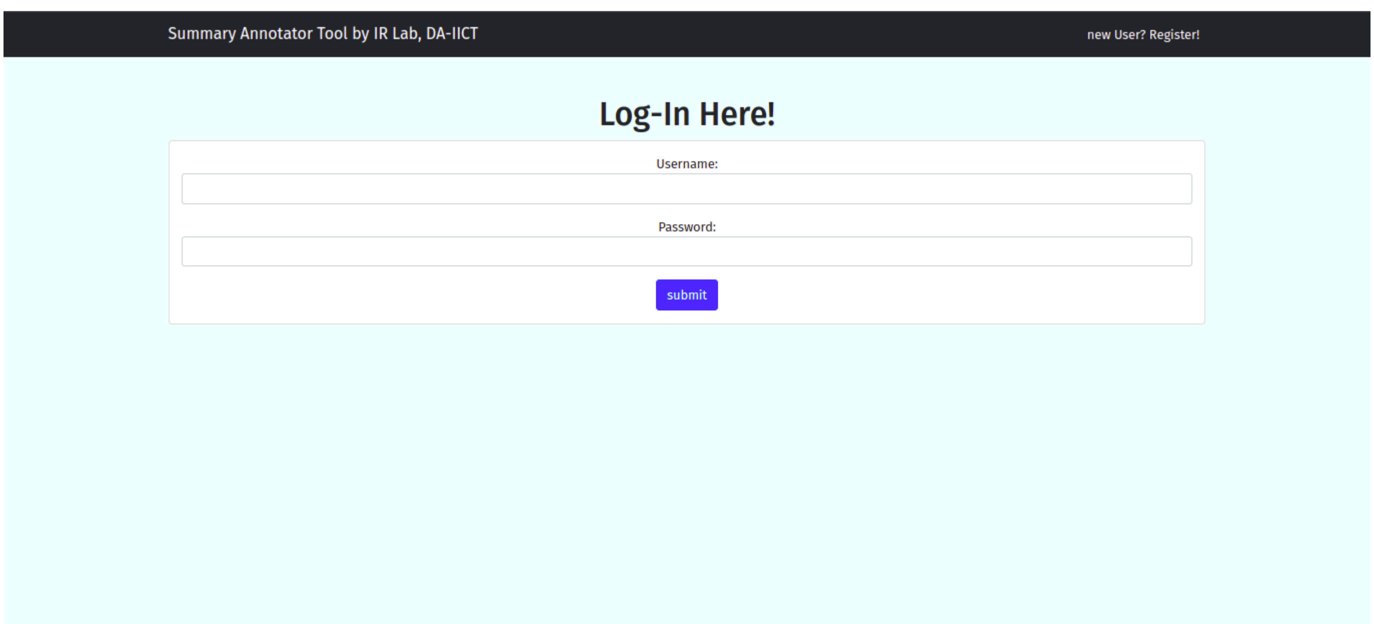

Figure B.1: Log-In Screen Prompted to the user

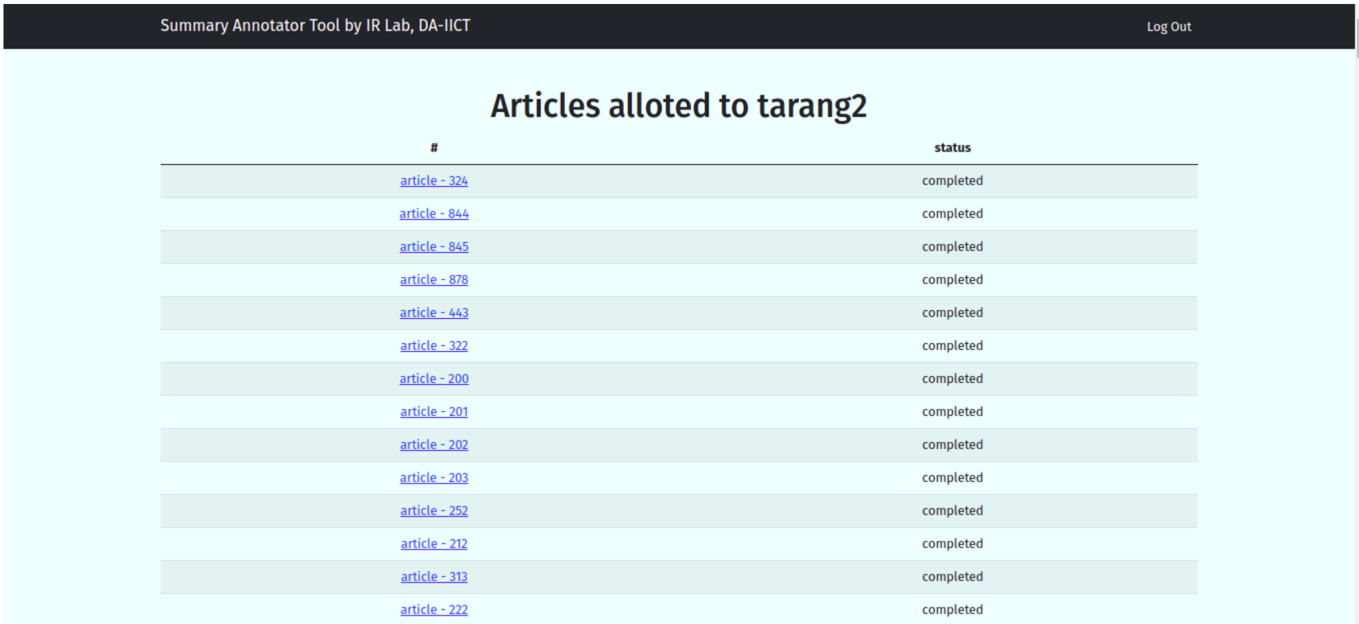

Figure B.2: Work Allocated to a particular user

Figure B.3: Annotation Sample allocated to a particular user

B.2 Admin's Side

```
(irlab) 10.100.59.162 — Konsole
(venv) irlab@irlab-xserver:~/SummaryAnnotatorTool$ python tool_utils.py help
help
supported functions:
        1. bulk_entry [csv_file_name]
        2. bulk_allocate [username] [email] [password] [n]
        3. bulk_allocate_to_existing [username] [n]
        4. export [filename]
        5. work_status
        6. help
```

Figure B.4: Functions supported at Admin's side: Create user, Bulk entry of tasks, allocation/deallocation of tasks, and check the work-status

```
(venv) irlab@irlab-xserver:~/SummaryAnnotatorTool$ python tool_utils.py work_status
work_status
           user   status
0       darshil    60/60
1       devansh  140/140
2        prahar    50/50
3       shradha    97/97
4       dhyanil  100/100
5      siddhant    50/50
6        kishan    30/30
7       krishna    30/30
8       tarang2  284/284
9        tarang      0/0
10         hari    60/60
11         sana  100/100
12  unallocated        0
```

Figure B.5: Work Status Information

CHAPTER C

Open Source Repository

We open-sourced a repository to replicate the results we presented. Along with the repository, we also open-source the dataset we built. This repository also serves as a "leader-board" to track the correlation of different metrics with human judgments. With this repository, we can track the performance of new models as they keep coming. Open-Source Repository: `https://github.com/TarangRanpara/EMFoS`