Systems and Resources for Telugu: Question Answering and Summarization

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering by Research

by

Priyanka Ravva
20172157
priyanka.ravva@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500032, INDIA
February 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"Systems and resources for Telugu: Question Answering and Summarization"** by Priyanka Ravva, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Manish Shrivastava

TO....

Parents, Family, Friends,

Research Group and the Institute.....

Acknowledgements

My journey in IIIT- Hyderabad has been a good experience. As I submit my MS thesis, I wish to extend my gratitude to all those people who helped me in successfully completing this journey. First of all, I want to thank my guide Dr. Manish Shrivastava for motivating me to pursue career towards the research and for giving opportunity to do internship and MS under his guidance. I sincerely thank Dipti Misra Sharma, Radhika Mamidi for their support in the lab during my MS. I thank my brothers Harish Yenala, Kranthi Kumar and my friends Teenu, Tinku, and Shiva for supporting me in my odds, taking care of me and keep on motivating me to reach a better place in my career.

I thank my friends Divija, Hiranmai, Ashok, Nirmal, Pavan, Lokesh, GopiChand, Shweta, Shirisha, DeviSree for being friendly with me throughout my stay at IIIT Hyderabad. I would also like to thank all my labmates Sunil, Hema, Mounika, Soumitra, Prashanth, Ganesh, Pruthwik, Krishna for the good interactions and fun work sessions. Special thanks to Lokesh Madasu and Gopichand Kanumolu for their assistance with the manual evaluations and all annotators for their support in creating the summarization dataset. I am thankful to K. Ravikanth (Faculty, RGUKT-Basar), J. Chakravarthi (Faculty, RGUKT-Nuzvid) and Satish Kumar Ch (Faculty, RGUKT-Srikakulam) for their constant and unyielding support.

I want to thank my parents, sister and brother for being my pillars of strength. They have been supportive throughout my journey and believed in me with all my decisions. Their unconditional love during my lowest and the highest points kept me going to achieve my goals. Without their tremendous support, this would not have been possible. Finally, I dedicate this thesis to my parents for their unconditional love.

Last, but not the least, thanks to IIIT community for giving me an inspiring environment and loads of opportunities to grow.

Abstract

Keywords: Natural language processing, Low resource language, Indian scenario, Telugu, Summarization, Abstractive Summarization Question-Answering, Sequence to sequence network with attention

Natural language processing (NLP) is a bridge between the computer and human interactions in their natural language. NLP has wide variety of applications such as machine translation, text summarization, question-answering, sentiment analysis, etc. All these applications have created huge impact in the society with different use cases such as chatbots, voice assistants (Alexa, Siri), recommendation systems (YouTube, Hotstar) etc. Most of these NLP applications are limited to few high resource languages like English. Notably, in an Indian scenario, where each state is having its own language, only 10% of people communicate in English. This motivates us to develop NLP systems and resources in Indian languages that will create a huge impact in multilingual Indian society. In this thesis, we created question-answering (QA) and text-summarization resources and systems in Telugu language.

The main aim of QA system is to provide an accurate and concise answer to the question asked by human in natural language. In this thesis, we created a question classification dataset which consists of 1037 samples and also explained the ambiguities, challenges involved in creating the dataset. We built an end to end pipeline for QA system and named it as AVADHAN. We performed comparisons between three different classifiers for the Telugu Question Classification (QC) module. QC will be helpful to reduce the search space while extracting the answer for the given query.

Text summarization is a way of obtaining short and precise summary from the given document of arbitrary length. In this thesis, we have proposed a pipeline that crowd-sources summarization data and then aggressively filters the content with automatic and partial expert evaluation. With this pipeline we have created TeSum: high quality human generated abstractive summarization corpus for Telugu. This corpus consists of 20329 high quality article-summary pairs and this is the first high quality and large abstractive summarization corpus in Telugu as per our knowledge. We have also explored different sequence to sequence neural networks on TeSum corpus and provided ROUGE scores.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural language processing (NLP) is a bridge for interactions between the computer and human language. NLP is a combination of linguistics, artificial intelligence and computer science. It is an emerging research field with many sub-fields such as text summarization, question answering, machine translation, sentiment analysis, and recommendation systems etc. These applications have shown significant impact in the society by making day-to-day human life easier.

However, building these NLP applications require large amount of processed(or annotated) text data and computational resources. And such processed text data is usually available only for few popular languages such as English. This limits the usage of NLP applications to specific community. To extend the benefits of NLP applications and increase its users across communities, there is a need to develop these NLP applications in a wider range of languages. This thesis builds the motivation to develop the NLP systems in Indian languages.

1.1 Need for Language-Specific Resources and Systems

India is a multilingual country having more than 1500 languages in which 22 are considered as official languages. The majority of these languages are derived from two language families: Dravidian and Indo-Aryan [1]. The south Indian languages are originated from Dravidian family and north Indian languages are originated from Indo-Aryan family. Most of the Indian languages are considered as low resource languages in NLP community that means these languages have limited or no processed text data. This motivates us to create the language specific resources and systems for Indian languages. In this thesis, we mainly focus on south Indian language Telugu. The Telugu language has more than 80 million native speakers. So If we develop the NLP systems in Telugu language it will benefit larger community. In this thesis, we build the two NLP applications such as question-answering system and summarization module.

1.2 Available Databases for Low Resource Languages:

In this section we present the statistics of low resource Indian language databases for different NLP tasks. We have discussed the detailed analysis of existing databases for summarization task and brief overview of existing Indian language corpus for other NLP tasks. Monolingual corpora [2],[3], parallel translation corpora [4], wordnet corpus [5], and named entity recognition corpus [6] are available for Punjabi, Hindi, Urdu, Marathi, Gujarati, Bengali , Odia, Assamese, Kannada, Telugu, Malayalam, and Tamil. Parallel transliteration corpora [7] and Parts of speech tagged corpus and chunk corpus [8], [9] is available for Punjabi, Hindi, Urdu, Marathi, Gujarati, Bengali , Kannada, Telugu, Malayalam, and Tamil. Dependency parsing corpus [10] is available for Hindi, Urdu, Marathi, Bengali , Kannada, Telugu, Malayalam, and Tamil. Text Classification corpora [2], [11] is available for Punjabi, Hindi, Marathi, Gujarati, Bengali, Odia, Kannada, Telugu, Malayalam, Tamil. Paraphrasing corpora [12] is available for Punjabi, Hindi, Malayalam and Tamil. Sentiment analysis corpora [11] is available for Hindi and Telugu. Question answering system corpora [13], [14], [15] is available for Hindi, Bengali, Telugu and Tamil.

1.2.1 Comparative Study of Existing Summarization Corpus for Telugu

We have performed a comparative study of existing Telugu summarization datasets and presented in Table 1.1. From the Table 1.1, the following observations can be drawn: 1. It is observed that the majority of these datasets contains less than 1000 text-summary pairs, which is a barrier in exploring the state-of-the-art neural network architectures. 2. Apart from XL-Sum [16], and MassiveSumm [17] no other data/code is publicly accessible. 3. These datasets were crawled from the web, and none of these datasets followed any quality assessment (human evaluation) criteria to assess the dataset quality. 4.We performed quality assessment on XL-Sum [16], and MassiveSumm [17] datasets in section 4.4.1.

1.2.2 Comparative Study of Non-Telugu Summarization Datasets

The Table 1.2 contains the stats of low resource summarization databases. Form the Table 1.2, the following observations can be drawn: 1. Summarization datasets are available for only few Indian languages. 2. The majority of the datasets related to the news domain consist of less than 100 samples and are publicly not accessible. Recently released multilingual abstractive summarization dataset XL-Sum [16] consist of corpora for 8 Indian languages as detailed in Table 1.3. Even though the XL-Sum dataset seems to promising contribution to the many low-resource languages, we have observed some limitations in the dataset quality, the detailed analysis on the limitations is presented in Chapter-4 section 4.4.2.

Reference	Type of summarization	Dataset size	Source	Categories	Techniques used
[18]	Extractive	1	News website	Politics	Frequency based approach
[19]	Extractive	360	Eenadu, Sakshi, Andhrajyothy, Namaste Telangana	Unknown	Heuristic based approach
[20]	Extractive	450	Eenadu, Sakshi, Andhrajyothy, Vaartha , Andhrabhoomi	Unknown	Keyword extraction approach
[21]	Abstractive	1	News website	News document	Keyword extraction approach
[22]	Abstractive	1	Unknown	Unknown	K-means clustering and Frequency based approach
[23]	Abstractive	8	Unknown	Sports	RNN + attention mechanism
[24]	Abstractive	30	Unknown	Biographies, Natural disasters, Reviews of products, Cultural events, Cricket	Template based approach
[25]	Abstractive	2000	Telugu news websites	Politics, Entertainment, Sports, Business, National	Seq2seq + attention
[16]	Abstractive	13205	BBC	National, International	Multilingual pre-trained model (mT5)
[17]	Abstractive	119282	News platforms	News documents	RNNs+Attention Transformers

Table 1.1: Comparative Study of Telugu Summarization Datasets

1.3 Available Systems for Low Resource Language Telugu

In NLP, text summarization, machine translation, question answering etc are considered major tasks. Limited work has been done towards these applications for low resource languages due to lack of resources. In literature, we have found some references for Telugu language systems such as machine translation [35], [36],[37], sentiment analysis [38], [39], [40], text-classification [41], speech-recognition [42], [43] etc. Further, in this section, we have provided detailed overview of available systems in Telugu language for question-answering and summarization task.

Reference	Language	Type of summarization	Dataset Size	Source	Categories	Techniques Used
[26]	Kannada	Extractive	18	Unknown	Arts, commerce, literature, homeopathy	Latent semantic analysis
[27]	Kannada	Extractive	20	kannada. webdunia.com	sports, religion, entertainment, literature, astrology	Keyword Extraction approach
[28]	Odia	Extractive	20	Unknown	Unknown	Sentence Ranking
[29]	Urdu	Extractive	20	Unknown	Scientific writings, economical items, sports	Sentence extraction approach
[30]	Hindi	Extractive	30	Unknown	Unknown	Rule based approach
[31]	Bengali	Extractive	38	Ananda Bazar Patrika	Unknown	Sentence Ranking
[32]	Malayalam	Extractive	50	Mathrubhumi	News	Heuristic based approach
[33]	Punjabi	Extractive	100	Pujabi Ajit news and www.likhari.org	News and stories	Keyword Extraction approach
[34]	Marathi	Extractive	634	EMILLE (Enabling Minority Language Engineering)	Unknown	Graph based method

Table 1.2: Comparison of low-resource extractive summarization datasets

Language	Size of the dataset
Bengali	10126
Gujarathi	11397
Hindi	88472
Marathi	13627
Punjabi	10627
Urdu	84581
Tamil	20276
Telugu	13025

Table 1.3: Data Statistics of Low-resource Abstractive summarization Datasets

1.3.1 Telugu Question-Answering Systems

For the question-answering (QA), a few dialogue based systems were explored [44], [45], [46], [47]. Most of these systems are limited to closed domain. In literature, we have not found any references for factoid Telugu QA systems and open-domain QA systems.

1.3.2 Telugu Text Summarization Systems

The last column of Table 1.1 summarizes the different methods used in literature for Telugu summarization. K-means clustering, frequency and ranking based approaches were explored towards Telugu extractive summarization. For abstractive summarization keywords based and keywords along with frequency based, IE rules with template based systems were implemented. For single document abstractive summarization, limited work has been done using seq2seq+attention models [25], where they has used scraped news dataset, headlines as summary that contains 2000 pairs. Multi-document abstractive summarization implemented using semantic similarity with the dataset of 8 samples (they did not provide the reference for dataset). Multilingual (44 languages, Telugu is one of them) abstractive summarization is implemented using pre-trained model (mT5) [16].

1.4 Challenges Involved in Creating Telugu Corpus for NLP Applications

In this thesis, we have created two different datasets in Telugu language for QA system and summarization task. Question-Answering system require a database that consists of question-answer pairs and question-label (based on the answer type) pair. The summarization task require a database that consists of article-summary pairs. Below we have mentioned some challenges involved in creating these datasets.

- **For question-answering system:**
  - Answer for a particular question can be dynamic
  - There is a possibility to have more than one answer for same question
  - Same question can be written in multiple ways by changing the word
  - Same question can be categorized into multiple classes based on the context of answer
- **For summarization task:**
  - There are no proper guidelines to create a high-quality summary of an article
  - There are no proper rules for quality assessment of summary

- Huge amount of high quality summarization data is required to get benefit of deep learning architectures which is very expensive and time consuming

1.5 Contributions of the Thesis

- We have build the question-answering system for low resource language Telugu named 'AVADHAN' using the manually created question-answer pairs. The corpus consists of 1037 pairs.
- We have proposed novel guidelines for Creation & Evaluation of summarization dataset. These guidelines can be used for any language to create a high-quality summary. The complete pipeline of database creation is explained.
- We have explored deep learning models for low resource language Telugu with the manually created summarization dataset.

1.6 Organization of Thesis:

The thesis is organized as follows: Chapter 2 presents an overview of existing question-answering systems and summarization systems for low resource languages. Chapter 3 explains the open-domain question answering system for Telugu language. Chapter 4 presents the proposed guidelines for manual summarization & evaluation and the complete process to construct the high-quality summaries. Chapter 5 explores different baseline architectures for Telugu abstractive summarization. Chapter 6 presents conclusions and a few ideas to future works.

Chapter 2

Literature Review

Natural Language Processing (NLP) is a subfield of artificial intelligence (AI). It helps machines process and understand the human language so that they can automatically perform repetitive tasks. Examples include machine translation, summarization, question-answering system, ticket classification, and spell check. The input to the NLP systems is text and output can be text or some other form based on the task. The flow diagram for NLP applications is shown in the Figure 2.1, the input text is converted into real valued vector representation and then this representation is processed by the model and model gives the output as text or class label based on the application. In the Figure 2.1, for the input "Who is the head of LTRC ?", the machine translation (English to Hindi) output is text and the QA system output is "Vasudeva Varma".

In this thesis, we are presenting two tasks question-answering and summarization for low resource language Telugu. Based on these two task the remaining chapter is organized as follows: Section 2 presents the literature review of question-answering system, summarization, and low resource language Telugu databases. Section 3 presents the overview of vector representations which are used in this thesis. Section 4 presents the brief introduction about simple feed forward classifiers and sequence to sequence classifiers. Section 5 presents the evaluation metrics for question-answering and summarization tasks.

2.1 Literature Review

Building high-level applications in NLP for low resource languages is a challenging task. Due to lack of resources many systems developed over the past few decades are limited to high resource languages like English. In this section, we discuss the work progress of QA system, summarization system and low resource language Telugu databases.

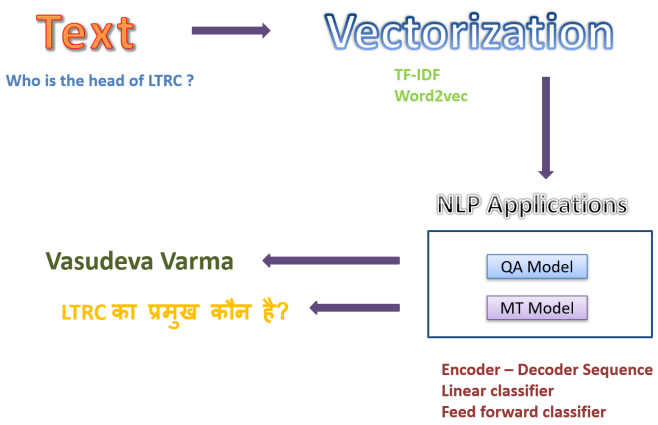

Figure 2.1: Overview of NLP Applications

2.1.1 Literature Review on Question Answering Systems

Humans are always eager to capture information about some topic or entity. A question-answering system helps users find the exact answer to a question expressed in natural language. A question-answering system provides concise, and accurate answers to user questions, rather than providing a set of related documents or web pages like most information retrieval systems do. Many automatic QA systems [48], [49] have been developed over the past few decades. Initially, the rule-based approaches were used as a prominent method to do the question answering [50] task. These approaches use semantics as decision trees to identify the appropriate answer to the given query. Manual creation of the rules was the main drawback of these systems. The statistical approaches like [51], [52] overcame the drawbacks of the rule-based approaches by predicting the suitable answers based on the huge chunks of available data. Later, it turned to the machine learning models [53] for predicting the answer which comes under knowledge-based question answering. As the information grows rapidly, the features get very expensive to process. As a result the processing speed decreases with the growth of data. To overcome this, deep learning concepts [54], [55], are applied to knowledge-based question answering to predict the correct answer for user query.

Zheng et al. [56] is a web-based QA system called as AnswerBus. It was implemented for five different languages (English, German, French, Spanish, Italian). This system retrieves the information from five search engines (Google, Yahoo, WiseNut, AltaVista and Yahoo News) and extracts the information by processing these retrieved documents. The authors mentioned the overall percentage of predicting exact answers for TREC-8's data set which consists of 200 queries is 70.5%. A state-of-the-art model LAMP [57] related to the web-based question answering system, which only takes the snippets returned by a particular search engine like Google. By taking only snippets LAMP reduces the time-consumption to retrieve the answer

compared to other state-of-the-art models. This research group collected snippets from the top 100 Google results for each query and applied a score function to retrieve the valid answers. They obtained the Confidence Weighted Score (CWS) of 0.60 and the average Mean Reciprocal Rank (MRR) score of 0.47 for LAMP system. Using the data set of 60 queries, [58] performed QA task for Hindi language. Authors particularly concentrated on 'where', 'when', 'how many' and 'what time' type of queries. These authors initially translated the Hindi language into Query Logic Language [59] using a tool named as BabelFish (AltaVista's translation tool). The overall accuracy of their QA system is 68%. Some other researchers [60] worked on factoid based QA systems by separating the data for only 5 types of categories (Location, Person, Name, Date, Others). They got the the accuracy of 62.11% with TREC (1999-2003) factoid data. To pick the exact answers from the extracted information, these authors [61] used paragraph indexing. One more question answering system 'LASSO' was proposed by using the paragraph indexing [61] to search for an answer by doing paragraph filtering and ordering.

In Indian context, QA systems are build for few Indian languages based on keywords, pattern matching and ranking of sentences to extract the answer [62],[63], [64]. Statistical and ML based QA systems are explored for few Indian languages [65],[66], [67], [68] , [69]. To the best of our knowledge, QA systems are not existed for Kannada, Tamil, Odia, Kashmiri, Manipuri, Konkani, Bodo, Dogri, Nepali, Sindi, Santai, Maithili languages. Specifically for Telugu, Bandyopadhyay et al. [44] implemented a dialogue based QA architecture on railway information data set. This database consists of 82 queries, in which 79 queries produce plausible answers. This approach exhibits the dialogue success rate of 83.96% and precision of 96.34%. Apart from this, there are few dialogue based QA systems for Telugu [45], [46], [47].

2.1.2 Literature Review on Text Summarization Systems

The text summarization task for low resource languages has been a long-standing problem in Natural Language Processing. There are two types of summarization abstractive and extractive summarization. The mechanisms involved in abstractive and extractive summarization differ significantly but have some common threads such as salience and coverage etc. In Indian context, the summarization task has been attempted using different approaches varying from statistical to linguistic-based and pure machine learning to hybrid methods.

Extractive Summarization Approaches: In extractive summarization, summary is created by selecting the and arranging the important sentences. The main challenge in extractive summarization is to effectively choose the important sentences and arrange them in proper order. To achieve this, early works for Telugu summarization use heuristic-based approaches [70], frequency-based and clustering approaches [18]. Due to the out-of-order extraction in the k-means extraction method, the summaries with the frequency-based approach makes more sense. The challenges of ranking the relevant sentences in the documents is usually addressed with the help of TextRank [71] and PageRank [70] algorithms. In addition, multi-document

summarization for Telugu has also been attempted [23, 72] . The majority of Indian language summarizers were built to perform extractive [21, 29, 31, 73, 74, 75, 76, 77] summarization due to ease in implementation.

Abstractive Summarization Approaches: Some of the abstractive summarizers were designed by using the information extraction methods [78] and automatic keyword extraction [20] by using the POS tags to generate the headlines. The summarizer uses components such as word cues, keyword extraction, sentence selection, sentence extraction, and summary generation module analyze the textual data to determine the key features of the summary. A Recent work on building neural abstractive summarization system [16] attempts Telugu summarization by directly scraping the text-summary pairs from BBC Telugu[1]. Unlike these baseline models [79, 80, 81, 82], authors of [16] have only fine-tuned the multilingual pretrained model (mT5).

2.2 Vector Representation

The input to any NLP applications is plain text but the machine can not process the raw text directly. As shown in the Figure 2.1, the first step in any NLP application is to convert the processed text into machine understandable format. This process is called text vectorization. There are two types of vectorization methods statistical and deep learning approach.

2.2.1 Statistical methods

TF-IDF: Term frequency-inverse document frequency (TF-IDF) vectorizer, is one of the popular approach for traditional machine learning algorithms, it helps to convert text into vectors. It is easy to calculate and understand. TF-IDF used to find the importance of a word is inversely related to it's frequency over all documents. TF gives us information about how often a word appears in a document, and IDF gives us information about the relative rarity of a word in a collection of documents. As TF-IDF compute the relative importance of a term based on a document, a search engine can use TF-IDF to help rank search results based on relevance, more relevant to a user with a higher TF-IDF scores.

2.2.2 Deep Learning Methods

Word2Vec: These embeddings can be trained using two neural networks Skip-Gram and Common Bag Of Words (CBOW). The CBOW model considers the input as window of surrounding words and try to predict the target (middle) word. The Skip-Gram model given an word, it predicts the window of surrounding words. Skip-Gram works well with small amount of training dataset and it gives better representation for the frequent words.

[1] https://www.bbc.com/telugu

Bidirectional Encoder Representations from Transformers (BERT): It is a transformer based deep neural network model, it converts words, phrases, sentences etc into a fixed dimension real valued vector. BERT embeddings are content dependent. That means it is possible to have more than one vector representation for the same word, based on context of the word in the sentence. For example, the BERT embeddings of the word 'bank' can be different based on the context.

2.3 Modelling Approaches

The input text is converted into vector representation then it is processed with a model that predict the output. The model can be machine learning or neural network based architecture. In this thesis, for the question-answering task we have explored simple machine learning algorithms like multi layer perceptron, logistic regression, and support vector machines. For summarization task, we have explored the sequence to sequence models such encoder decoder with attention mechanism.

2.3.1 Machine Learning Classifiers

- Multi Layer Perceptron (MLP) a simple feed forward architecture which contains one or more hidden layers. While a single layer perceptron can only learn linear functions but a multi-layer perceptron learns non-linear functions too. By increasing the number of hidden layers, it is possibility to get a more accurate model.

- Logistic Regression (LR) used to assign labels for given input query depending on its context observation. LR can be applied to multi-class problems by using the multinomial type, which is called multinomial logistic regression.

- SVM is used for both regression and classification task but mostly for the classification problems. The main aim of this algorithm is to find the maximum margin hyperplane in an N-dimensional space that uniquely classifies the data points. This classifier gives significant accuracy with less computation power.

2.3.2 Deep Learning Classifiers

Sequence to sequence classifier maps variable length input sequence to variable length output sequence. The basic form of this classifier consists of an encoder, attention, and decoder modules. The encoder and decoder are stack of RNN/LSTM/Transformer layers. In general encoder has many layers compared to decoder. The detailed description of the encoder-decoder architecture and the basic modules used for encoder and decoder are given below.

2.3.2.1 Encoder Decoder With Attention

An encoder converts the variable length input into a fixed length output. The encoder processes each token of an input sequence and accumulates the information. After processing the entire input sequence, the accumulated information for each taken is scaled using the attention mechanism based on the importance of the token. Then the scaled context information of tokens is averaged over token sequence length. This represents the entire sequence context information into a fixed dimensional representation. This fixed length information is passed to decoder which will help the decoder to make accurate prediction in a auto regressive manner. The decoder generates the output sequence token by token until the end token. The encoder-decoder with attention is shown in the Figure 2.2.

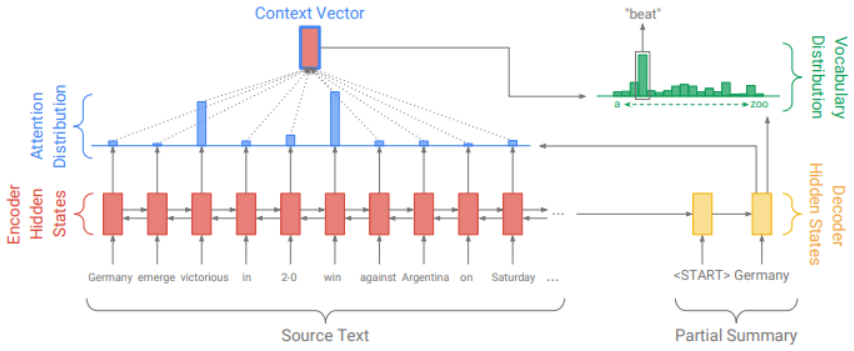

Figure 2.2: Encoder-Decoder Architecture

2.3.2.2 Building Blocks of Encoder and Decoder

In encoder and decoder network we can use either recurrent neural network (RNN) or long short term memory network (LSTM) or Transformer as building blocks based on the requirements like performance, computational complexity.

- **RNN** process the input sequentially by carrying the previous token information during processing the present token. Sometimes we need to look at only the recent relevant information in order to perform the present task. In this case the gap between relevant information and point where it is required is small, RNN can capture that information. In some cases the gap between the relevant information and the point where it required is more then RNN unable to connect the information. This problem is solved by LSTMs.
- **LSTM** is a kind of RNN, and it learns long-term temporal dependencies through gating mechanism. LSTMs main key strength is in the cell/memory state. The gating mechanism

control the information flow either adding or removing information to the cell state. It has three gates input, forgot, and output. The input gate calculates the current token importance and forgot gate helps in forgetting the information which is not important from the previous tokens. The output gate decides how much information has to send to next token.

- **Transformer** captures sequential information with simple self-attention and feed forward network. The self attention captures relative information in the sequence with respect to tokens. Transformers are computationally less expensive and fast as compared to the RNN and LSTM.

2.4 Evaluation Metrics

The performance of the question-answering system is evaluated using the accuracy and summarization system performance can be evaluated using ROUGE score. Below sections give the brief overview of these metrics.

2.4.1 Question-Answering System Evaluation

The sequence classification (QA system) task is evaluated using the simple accuracy. Here, we matches predicted answer with reference answer. If it matches with reference answer then it count as correct sample. The formula is presented below.

$$Accuracy = (\frac{Numberofcorrectsamples}{Totalnumberofsamples}) * 100 \tag{2.1}$$

2.4.2 Summarization System Evaluation

The sequence to sequence summarization task is evaluated using the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metric. ROUGE is a set of metrics used in NLP for evaluating automatic summarization and machine translation applications. This metrics compares the system generated summary or translation with a reference or set of references (human-generated) summary or translation. ROUGE metric has the following 5 metrics: ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S, ROUGE-SU. The detailed description for these five metrics is given below.

- **ROUGE-N:** It matches n-grams between the system generated summary and reference summary(s).

    1. **ROUGE-1** refers to the overlap of unigram (each word) between the system and reference summaries.

2. **ROUGE-2** refers to the overlap of bigrams between the system and reference summaries.

- **ROUGE-L** is longest common subsequence (LCS) based statistic. The longest common subsequence covers both sentence-level structure similarity and identifies longest co-occurring in sequence n-grams automatically.
- **ROUGE-W** is a weighted LCS based statistics that favor consecutive LCSes.
- **ROUGE-S** is a skip-bigram based co-occurrence statistics. Skip-bigram is any pair of words in their sentence order.
- **ROUGE-SU** is a Skip-bigram plus unigram-based co-occurrence statistics.

Chapter 3

AVADHAN: System for Open-Domain Telugu Question Answering

Question answering (QA) system is one of the most trending application in Natural Language Processing (NLP) research. QA system is fast growing and interesting research area. It combines the research fields of Information Retrieval (IR), Information Extraction (IE), and NLP. The main aim of QA system is to provide an accurate and concise answer to the question asked by human in natural language rather than just providing the list of question related documents like a search engine. QA systems are classified into two types, closed domain question answering system and open domain question answering system. Closed-domain QA systems only handles the questions within a specific domain, whereas an open-domain QA systems handles almost all sort of questions irrespective of domain. Type of questions can be of different form namely factoid, definition, list, and description. In this thesis, we are dealing with an open-domain question answering system with factoid-type questions for the Telugu language.

To build an open-domain QA system we need a proper dataset which contains question-answer pairs. As this is the first attempt to build the Telugu QA system, we couldn't find any open source dataset. This indicates the requirement of creating a new database which contains question-answer pairs in Telugu language. This chapter discuss about the creation of the database, QA system architecture named AVADHAN and performance analysis of QA system with different classifiers.

3.1 Dataset Preparation for Telugu QA system

This section discuss the database preparation details such as collection sources, scope of the database, and labelling methods.

3.1.1 Corpus Collection

Question answer pairs can be created manually by writing or else we can scrap these pairs from the websites. In this work all the question-answer pairs are collected from these websites [1] [2] [3] using web scraping techniques in python. The data set collection was started in August 2017. The main reason behind specifying the exact time for the corpus preparation lies in answering dynamic queries or queries dependent on time. While preparing this kind of QA dataset, many times, there is a possibility of change in target answers with respect to time (like presidents of the countries, sports records, population records, etc). In this thesis, we limited the scope of the database to static queries which does not change with respect to time. The final dataset contains 1037 Telugu queries.

3.1.2 Database Labelling Approach

Question-Answering system provides the most appropriate answer to the question in natural language that is extracted from the large set of documents. As the number of documents increases from which the answer is to be extracted, it is difficult to choose the most relevant one. Also, QA system has to deal with multiple category questions which are based on the type of answer. As the multiple category questions increases, the search space also increases, this may hinder the performance of QA system or demands huge amount of training data to build QA system. These problems can be solved by reducing the search space of questions by providing the category of question to the QA system. The module that provides category of question is called question classification (QC) module. We can use any pre-trained QC module to categorize the question. But in literature we did not find any QC module for Telugu language.

Question classification module can be of rule based or machine learning based module. Machine learning model works better than rule based methods. The machine learning models demands large amount of training data compared to rule based approach. Training of QC module requires the question/query and corresponding label pair. In literature, as per our knowledge this type of labeled data is not available for Telugu language. This makes Telugu QC a challenging task and also creates a need for developing the QC database for Telugu language. As discussed in section 3.1, questions collected from the websites are used for labeling. The labeling of the questions have done with the help of three annotators. The annotators read the each question and tag the corresponding label based on the answer of the question. The final QC corpus contains seven categories of questions based on answer type. The seven query categories are Person, Location, Number, Organization, Time, Date, and Percentage.

[1] https://mympsc.com/appsc

[2] https://upscgk.com/APPSC-gk

[3] http://services.indg.in/online_quiz/index_te.php

Answer Category	Query type	Outcome
పేరు (PERSON)	ఆనంద్ మఠ్ పుస్తకం రాసినది ఏవరు ? (Who wrote the book 'Anandamath'?)	బంకీమ్ చంద్ర చటోపాధ్యాయ (Bankim Chandra Chattopadhyay)
సంస్థ (ORGANIZATION)	ప్రపంచంలో అత్యంత లాభదాయక సంస్థ ఏది? (What is the world's most profitable company?)	అరంకో (Aramco)
నగరం (LOCATION)	ప్రపంచ ఆరోగ్య సంస్థ యొక్క ప్రధాన కార్యాలయం ఎక్కడ ఉంది? (Where is the headquarters of the World Health Organization?)	జెనీవా (Jeneva)
సంఖ్య (NUMBER)	మానవ శరీరంలో ఎన్ని ఎముకలు ఉంటాయి ? (How many bones are there in the human body?)	206
తేదీ (DATE)	ప్రతి సంవత్సరం ప్రపంచ పర్యావరణ దినోత్సవం ఎప్పుడు జరుపుకుంటారు? (when is the world environment day celebrated every year?)	జూన్ 5 (June 5)
సమయం (TIME)	చంద్రుడు భూమిని ఒకసారి చుట్టిరావడానికి ఎన్ని రోజులు పడుతుంది? (How many days does it take for the moon to encircle the earth?)	27 రోజులు (27 Days)
శాతం (PERCENTAGE)	సముద్రపు నీటి యొక్క సగటు లవణీయత ఎంత? (What is the average salinity of seawater?)	3.5 శాతం (3.5%)

Table 3.1: Categories of queries based on the type of answer

The seven query categories with examples are shown in Table 3.1. The Table 3.1, contains both Telugu and corresponding English translated question-answer and the corresponding question category in column wise manner. There are some challenges involved in categorizing the queries in Telugu language. For example, in some cases single word of Telugu language can fall in more than one category like 'యమునా' (Yamuna), which can be either person name or a river name based on the context. While creating the data set, we came across many special category types of queries. Some of them are mentioned in Table 3.2 and we named these

Type of query	Query type	Outcome
More than one answer possible query	2014 లో నోబెల్ శాంతి బహుమతి ఎవరికి వచ్చింది? (who got nobel peace prize in 2014?)	కైలాష్ సత్యార్థి, మలాలా యూసఫ్‌జాయ్ (Kailash Satyarthi, Malala Yousafzai)
Miscellaneous type Query	మీరు సరిగ్గా ఏమీ చేయలేరా? (Can't you do anything right?)	Rhetorical question
Time bounded Query	హాంగ్ కాంగ్ యొక్క ప్రస్తుత జనాభా ఎంత? (what is the current population of hong kong?)	73.9 లక్షలు (73.9 lakhs)

Table 3.2: Special types of queries based type of answers

confusing queries as special category. This corpus consists of total 100 special queries which do not belong to above mentioned seven categories. For example, in Table 3.2, the first query signifies one such scenario of more than one possible answers and the second one deals with a miscellaneous query, which is rhetorical[4] (which needs some discussion to decide the exact answer), the third query is a time-bounded one, in this case exact answer changes with respect to time.

3.2 Question-Answering Model Pipeline

In this section, we discuss the QA system architecture and it's main components. The QA system takes input as query and returns the corresponding answer. We named this as AVADHAN an end to end pipeline from query to answer. The AVADHAN architecture is show in Figure 3.1. The model implementation starts with the given user query, which is in Telugu language. First, this input Telugu query is translated into English and then relevant information is extracted from search engine. To do this, Python packages urllib2[5] and Google translate API[6] were used.

The AVADHAN system has three main components as shown in Figure 3.1. The detailed description of the components are discussed below.

1. Information retrieval

[4]https://literarydevices.net/rhetorical-question/

[5]https://docs.python.org/3/howto/urllib2.html

[6]https://py-googletrans.readthedocs.io/en/latest/

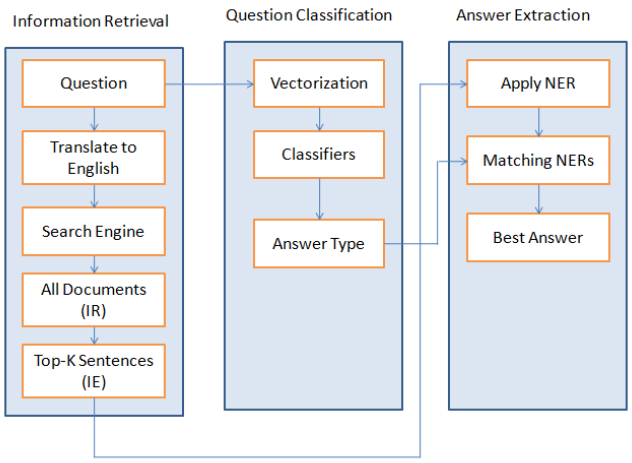

Figure 3.1: AVADHAN Architecture

2. Question classification

3. Answer extraction

3.2.1 Information Retrieval

The information retrieval (IR) system extracts relevant information of input query from the search engine. In this work we have used "Bing" search engine. The IR module takes translated English query as input and gives search results. In general, extracted search engine information can be considered as unstructured data. This unstructured information is converted into structured form using web scraping methods. To achieve this, Python libraries BeautifulSoup[7], urllib2 were used. Precisely, urllib2 is used for fetching Uniform Resource Identifier (URLs) and BeautifulSoup is to extract the data from the web pages. To avoid the noise in the data, we considered the top 10 URLs with the most relevant context for the query. We also extracted the useful meta information from each URL, which was attached with HTML tags like headings, paragraphs, tables, and images, etc. If any incomplete sentences or meaningless heading were found, they were ignored. After observing all the cases, suitable sentences are taken from multiple documents and stored in one document.

Further we process the above created document for getting more precise information. For this, we used a ranking methodology to find out the important sentences with in the document. This rank-based approach, provides better search speed by removing the noisy information at the prediction phase. We used cosine similarity matrix method for ranking the top K-sentences useful for the query which can the give the accurate answer. The cosine similarity method works on the vector data. So we convert the sentences into real valued embedding vectors. The

[7]https://www.crummy.com/software/BeautifulSoup/bs4/doc/

similarity is obtained by measuring the dot product between the two vectors. The dot product value lies between -1 and 1. If the dot product value is +1 means two sentences are very similar and if the dot product value is -1 means both the sentences are very different. The similarity can be obtained by measuring the angle between two sentences. The dot product value and angle are inversely proportional to each other. The dot product equation and angle between two vectors can be calculated using the equations 3.1 and 3.3.

$$\overrightarrow{a} \cdot \overrightarrow{b} = \| \overrightarrow{a} \| \cdot \| \overrightarrow{b} \| \, cos\theta \tag{3.1}$$

$$cos\theta = \frac{\overrightarrow{a} \cdot \overrightarrow{b}}{\| \overrightarrow{a} \| \cdot \| \overrightarrow{b} \|} \tag{3.2}$$

$$\theta = cos^{-1}\left(\frac{\overrightarrow{a} \cdot \overrightarrow{b}}{\| \overrightarrow{a} \| \cdot \| \overrightarrow{b} \|}\right) \tag{3.3}$$

3.2.2 Question Classification

Information retrieval module extracts the relevant information corresponding to the query. The relevant information is passed through QA system to get the answer for query. In general, instead of processing all relevant information, the search space can be reduced by identifying the answer type for the query. This can be done using question classification module. To get the answer type for a user query, initially we have to classify the query into predefined Named Entity Recognizer (NERs) classes as defined in Table 3.1 and Table 3.2. This predicted classes will play a major role in finding the answers accurately. In literature, we did not find any QC module for Telugu data. Finding the classifier which works better for Telugu is even more challenging task. So we have trained different classifiers like Logistic Regression (LR), Multi Layer Perception (MLP) and Support Vector Machine (SVM) using labeled data as described in section 3.1.2.

The input to the QC classifier is vector representation of query. Here, we have considered vector representation as Term Frequency Inverse Document Frequency (TF-IDF). Each query is considered as a document and each word as a term. TF-IDF representation is transformed version of text into a meaningful representation of numbers. It is the product of term-frequency (tf) and inverse document frequency (idf). Let C = number of times word (W) occurs in a document, N = total number of words in document, X= total number of documents, S = number of documents containing word (W). Below equations 3.4, 3.5, and 3.6 represents term-frequency, inverse document frequency and TF-IDF respectively.

$$tf(W) = \frac{C}{N} \tag{3.4}$$

$$idf(w) = log(\frac{X}{S}) \tag{3.5}$$

$$tf - idf(w) = tf(w) * idf(w) \tag{3.6}$$

The TF-IDF is given as input to different classifiers. In MLP classifier, we have considered one input layer, four hidden layers, and one output layer, with 'stochastic gradient descent' optimizer and tanh activation. In case of LR classifier used multi-class as 'multi-nominal' with 'stochastic average gradient' optimizer. Another classifier SVM , taken multi-class as 'one-vs-rest', with loss as 'squared_hinge'. Total labelled corpus is divided into train and test data in the ratio 80:20.

3.2.3 Answer Extraction

Question-Answering system takes inputs from both information retrieval module and question classifier. QC classifiers gives type of answer which is also known as named entity . Named entity recognition (NER's) [83], [84], [85] are popularly used in information extraction to recognize the named entities and categorize into various predefined classes. Identified NERs from QC module are applied on each of the top K-ranked sentences to extract the same category of answer type with the help of Spacy[8]. If more than one possible answers were obtained for any particular query, the best answer will be selected based on the frequency of occurrence. At the final step, the obtained answer will be converted into Telugu language with the help of Google Translate API.

3.3 Results & Discussion

In this section, we discuss the experimental setup, performance of QA system, and some important observations based on the exact match and partial match of answer to the query.

3.3.1 Experimental setup

All the experiments are performed with labeled corpus as described in section 3.1.2. The labeled corpus consists of total 1037 pre-processed queries which are divided into seven predefined categories. The predefined categories are mentioned in the Table3.1. More than 50% of corpus consists of LOCATION, PERSON and NUMBER categories. Data set split into train and test queries into 80:20 ratio.

[8]https://spacy.io/usage/linguistic-features#named-entities

3.3.2 Performance of question-answering system

Initial experiments are carried out to check the number of sentences required to fetch the correct answer. The results are depicted in Table 3.3 for different classifiers and for different K (number of sentences) values. It also contains results for exact match and partial match. Exact match means predicted query answer is exactly matched with ground truth and partial match means part of the predicted answer is matched with the ground truth answer.

Classifiers		K=10	K=20	K=30	**K=40**
SVM	EM	21.9	25.9	27.8	**31.6**
	PM	46.7	58.4	61.5	**68.5**
LR	EM	21.1	25.7	29.4	**31**
	PM	46.3	57.5	63.6	**66.6**
MLP	EM	21.5	24.5	29.4	**30**
	PM	47.3	57.4	62.8	**67**

Table 3.3: Overall performance of AVADHAN (in terms of %) by varying the number of sentences, EM: Exact Match, PM: Partial Match

Category	Number of samples	Exact match			Partial match		
		C1	C2	C3	C1	C2	C3
Location	353	34.6	35.7	36.4	70	72	72.3
Person	317	18.9	20.8	20.5	68.8	66.4	68.5
Number	170	44.1	44.7	42.6	58.2	59.4	56.8
Date	125	28	32.8	37.2	80	76	87.6
Organization	29	27.6	17.2	20	37.9	31	26.7
Percentage	25	28	26.1	24	40	39.1	48
Time	18	22.2	5.6	18.8	55.6	61.1	75
Overall	**1037**	**30**	**31**	**31.6**	**67**	**66.6**	**68.5**

Table 3.4: Accuracy (in terms of %) of AVADHAN for individual categories with K=40

The Table 3.3 contains accuracy results for different K values (10,20,30,40). It is observed that as K increases the performance is also improving and we got best results for K = 40. Further increasing the value of K, significant improvement in the performance is not observed. For small values of K, less number of sentences may increase the chances of ties among candidate answers because we are following frequency based answer extraction. We observed better predictions as K increasing. For large values of K, there may be a chance that some sentences

are irrelevant that eventually decreases the accuracy. Further experiments are conducted with K=40 sentences. It is also observed, the accuracy is almost halved from partial match of answer to exact match of answer to the query. Further analysis on partial and exact match is discussed in the below section with few examples.

In case of classifiers, SVM, LR, and MLP has almost equal performance with 1% change among classifiers. The overall accuracy of SVM is slightly better than both MLP and LR classifiers, as MLP and LR works better with a very large data set. Since, we are using less than 1500 queries data (because of data scarcity) MLP and LR are not able to produce better results. In general, SVM performs better with lots of features that are naturally on the same scale and fewer data points. It also uses a subset of training points as support vectors, these are used to get the clear margin of separation between two classes. SVMs are efficient and generates a best classification as they obtain the optimum separating surface, which has a good performance on previously unseen data points. Overall the MLP, LR, SVM classifiers obtains the accuracies of (30% , 67%), (31%, 66.6%) and (31.6%, 68.5%) for both exact match and partial match scenarios. The Table 3.4 contains the results for each category for exact match and partial match for different classifiers. In Table 3.4, first column represents category, and second column represents the number of queries for each category, and third to sixth column contains accuracy results for different classifiers.

From the Table 3.4, it is observed that the best accuracy is 68.5%. After doing multiple experiments still the best performance is limited to 68.5%. This is due do some issues related Telugu language. For example, one prevalent issue that can be seen in Telugu such as the choice of word changes with respect to nativity, social distance and the social status (relation) between the speakers. This is demonstrated with the below sentences. For the English sentence "How much Money Do You Need" there can be multiple ways to write it in Telugu by taking the different word choices for word Money.

1. ఎంత సొమ్ము కావాలి? (How Much Money Do You Need?)
2. ఎంత డబ్బు కావాలి? (How Much Money Do You Need?)
3. ఎంత ధనము కావాలి? (How Much Money Do You Need?)

3.3.3 Observations on Exact Match

In case of exact match, we look for an exact match with the specific answer. Most of the times obtaining the exact answer in Telugu context might be a difficult task for some category of queries. This might hamper its accuracy and this was experimentally observed in Table 3.4. For the following categories TIME, PERSON and ORGANIZATION the accuracy percentage is very low because of the uncertainty involved in the answer context. The uncertainty involved in those categories is explained with the examples in the below Table 3.5.

Query	Correct Answer	Predicted Answer	Possible Correct Answers
"ఆలిస్ ఇన్ వండర్ల్యాండ్" - పుస్తక రచయిత ఎవరు? ("Alice in Wonderland" - Who is the author of the book?)	లెవిస్ కారోల్ (Lewis Carroll)	లూయిస్ కారోల్ (Louis Carroll)	లెవిస్ కారోల్ (Lewis Carroll), లూయిస్ కారోల్ (Louis Carroll)
భారతదేశం యొక్క క్షిపణి మనిషి ఎవరు? (who is the missile man of India?)	ఎ. పి. జె. అబ్దుల్ కలాం (A. P. J. Abdul Kalam)	అవూల్ పకీర్ జైనులాబ్దీన్ అబ్దుల్ కలాం (Avul Pakir Jainulabdeen Abdul Kalam)	1. ఎ. పి. జె. అబ్దుల్ కలాం (A. P. J. Abdul Kalam), 2. డాక్టర్. అబ్దుల్ కలాం (Doctor. Abdul Kalam), 3. అవూల్ పకీర్ జైనులాబ్దీన్ అబ్దుల్ కలాం (Avul Pakir Jainulabdeen Abdul Kalam)

Table 3.5: Special Case Examples with Exact Match Method

The first row of Table 3.5 is an example of PERSON category. The predicted answer (Louis Carroll) for the first row query has very small difference with respect to original ground truth (Lewis Carroll) but it does not suitable for exact match. The irregularities like prefixes and suffixes added to the NAME or the affiliations added to the NAME category also decreases the accuracy for this category. The second row of Table 3.5 is query of special category (more than one answer). The predicted answer for second row query is Avul Pakir Jainulabdeen Abdul Kalam and the ground truth answer is A. P. J. Abdul Kalam.

The above mentioned issues are the major reasons for obtaining less accuracy for exact match case in Telugu question answering task.

3.3.4 Observations on Partial Match:-

In the case of partial match, prediction of the final answer is decided based on the threshold value. In our experiments, the threshold is fixed as 0.7. Precisely, if the predicted answer has partial-match-score more than or equal to the threshold value, then it is considered as plausible answer. Because of this reason partial match methods give better results compared to exact match cases. To find the partial match, we used the SequenceMatcher[9] library in Python. In this approach, ORGANIZATION and PERCENTAGE answer categories produced low accuracy because of the ambiguity in data set related to that category.

3.4 Conclusions

As an initial move towards Telugu question-answering system, this work broadly explained the challenges and uncertainties involved in creating the query data set for Telugu language. We build the question classifier that categories queries based on the answer of query. We named this

[9]https://docs.python.org/2.4/lib/sequencematcher-examples.html

end to end question-answering system from query to answer as AVADHAN. The AVADHAN system performance is evaluated by changing the number of sentences required for information extraction with different classifiers. We concluded that with support vector machine classifier, AVADHAN produces better accuracy compared to MLP and LR. We can further to reduce the time consumption for retrieving the best answer by directly taking the Google snippets data. In future, AVADHAN could also be expanded to multilingual open domain question answering with both static and dynamic data sets.

Chapter 4

TeSum: Human-Generated Abstractive Summarization Corpus for Telugu

Summarization is a natural language processing (NLP) task that requires, given a document of arbitrary length, a summarizer to return a shorter, relevant subset of the input for a specific purpose. There are two types of summarization abstractive and extractive. In abstractive summarization the summary is novel and creative which does not contain the exact sentences from the article. In extractive summarization the summary is arranged with important selected sentences of the article that precisely represents the article. A number of recently released datasets for summarization, scraped the web-content relying on the assumption that summary is made available with the article by the publishers. While this assumption holds for multiple resources (or news-sites) in English, it should not be generalised across languages without thorough analysis and verification. We look at a particular Indian language, Telugu, for which such datasets are available as XL-Sum [16] and MassiveSumm [17] which have been collected from various sources. Both rely on above mentioned assumptions. Even with a casual observation, we find that these assumptions, and therefore these datasets do not stand up to the test.

Dataset creation for any task is an expensive and complex activity. With the increased demand for data for deep-learning models, it is often infeasible to create datasets which reach the desired sample counts. It then does make sense to make do with data collected "from the wild". It is our opinion that such collected data, while useful, should also be subjected to quality control. At the same time, we should adopt pipelines which can establish a balance between quality control and cost. This is especially critical for Low Resource Languages which need to make do with low sample numbers. We find that there is a urgent and immediate need for a dataset and a dataset creation methodology which stays true to the essence of the task of summarization as defined in [86]. Such a dataset must be created with human involvement and thorough evaluation. Expert human annotation for summarization is definitely an expensive task, and can not be done on huge scales. But with this work, we show that even with a crowd sourced summary generation approach, quality can be controlled by aggressive expert informed filtering and sampling-based human evaluation. We propose a pipeline that crowd-sources

summarization data and then aggressively filters the content via: automatic and partial expert evaluation. Using this pipeline we create a high-quality Telugu Abstractive Summarization dataset TeSum which we validate with sampling-based human evaluation. We also provide baseline numbers for various models commonly used for summarization.

4.1 Crowd-Sourced Corpus Creation

We propose a crowd sourced summary creation phase followed by a curation phase by trained experts. We work with 347 "creators" and 3 expert "raters" for this task. The "creators" are provided with specific guidelines to ensure the quality of generated content. The content is then aggressively filtered to retain high quality article-summary pairs. Human experts then evaluate a subset of the collected article-summary pairs to remove substandard tuples.

4.1.1 Source

We scrape Telugu news sites for source articles, under fair usage policy and divide them into sets of 50 articles each. The copyright of the original articles remains with the original authors/publishers of the articles. We release TeSum dataset as a list of URLs and summary pairs. These articles are then processed to remove HTML tags, non-Telugu content and common irrelevant phrases (article dates, city names etc.).

4.1.2 Manual Summarization

The human summarization task is posed as a crowd-sourcing activity. Each HIT (Human Intensive Task) for a creator consists of 50 news articles set created earlier. The creator is expected to create summaries for all the articles in the HIT following the guidelines given below. Each HIT submitted by the creator undergoes thorough automatic and human evaluation steps in order to ensure quality based on a criteria which maintains the essence of the task of summarization. The creator needs to ensure that:

1. Relevance: All or most of the relevant information contained in the article should be present in the summary.

2. Readability and Coherence: The summary should be coherent, readable and free of any grammatical errors.

3. Creativity: The summary should have novel sentential and phrasal structures.

The human summary creators were given the guidelines presented in Section 4.1.2.1 based on the above 3 properties.

4.1.2.1 Guidelines for Abstractive Summary Creation

Summary creators were instructed to carefully follow these guidelines and write one abstractive summary per article.

1. **Relevance and Coverage:** All the pertinent information conveyed in the source article should be captured in summary while discarding any irrelevant information. Redundant information or information unrelated to the major topic of the article may be considered irrelevant.

    - **Missing important information:** A summary has to cover all the important aspects of the original article.
    - **Including irrelevant information:** A summary should not include any irrelevant information. No personal opinion(s) or non-factual details should be included.
    - **Redundant information:** Summary should not contain any repetitive phrases/sentences.

2. **Readability:** If the summary is understandable by a native speaker without looking at the source article, it is considered "Readable". Bad grammar, pronouns that cannot be resolved within the summary, and unnatural sentential/phrasal structures would make the summary difficult to understand. Also, creators are instructed that the summary should stand as an independent article, and the reader should not need the original article to understand it fully.

    - **Disjoint sentences:** While paraphrasing, sentences should be joined in such a way that the composite sentence must be meaningful.
    - **Anaphora issue:** In summary, pronouns should be used only after the antecedent has appeared at least once.
    - **Disordering of sentences:** The summary should be coherent to convey the proper context of the original article.
    - **Not readable:** The summary should be free from any syntactic and semantic errors.

3. **Creativity:** Since this is an abstractive summarization task, we require the summaries to have novelty in terms of sentential structures such as lexical choices (vocabulary used, is other than the given article), phrasal constructions, and sentence formations.

    - **Missing novel sentence structure:** The summary should contain novel sentence structures (using some novel words) compared to the original article.
    - **Lengthy summary:** The summary should be a new shorter text that conveys the most crucial information of the original article.

- **Sentence level summary:** The summary should not be created by just altering words/phrases in individual sentences.

4.2 Corpus Curation Process

As expected with any crowd-sourced text annotation task, the summaries generated by the creators had a wide variety of errors. As shown in the guidelines, we have not instructed the creators to create their summaries within a pre-specified character or word limit. This is done to ensure that the creators do not feel restricted while writing the summaries in order to fit within a pre-specified limit. An artificial limit to the length of the summary at the creation phase might introduce unnatural structures or phrasing in the sentences of summaries. Instead, we decide to filter these generated summaries on the basis of multiple automated criteria after the summaries have already been created. The crowd-sourcing activity resulted in a collection of 92941 article-summary pairs. These articles were then filtered based on the following two stages.

4.2.1 Automatic Filtering

Even with basic sanity checks at the crowdsourcing stage, we encounter a large number of errors in these submitted HITs. These article summary pairs need to be filtered out in order to maintain the high quality of the dataset.

1. *Remove Empty:* We remove any pairs where either the summary/article or both are empty.

2. *Remove Duplicates:* Duplicate pairs and duplicate summaries were removed. We do not want duplicate article-summary pairs. Two distinct articles should not have the same summary. We find it is unlikely that two distinct articles would share the same summary, therefore we also remove the pairs which share a common summary.

3. *Remove Prefixes:* We remove all prefix cases, that is any pair where the summary is just the first few sentences of the article. We should note that though MassiveSumm has claimed to follow steps $1-3$, we find in Table 4.1 that applying these steps to MassiveSumm, a large volume of their samples still fell in to these categories. Duplicate summaries case holds true for XL-Sum also.

4. *Remove Article Length $<$ 4 Sentences:* We removed 4452 pairs with less than 4 sentence articles.

5. *Remove Article Length $<$ 40 Tokens and / or Summary Length $<$ 10 Tokens:* Very small article lengths are not indicative of the general distribution of news article data.

	XL-Sum	MassiveSumm	TeSum
Dataset Size	**13025**	**119282**	**92941**
Empty	2	5579	2
Duplicate Pairs	0	10456	515
Duplicate Summary	141	2698	135
Prefixes	3	30741	1330
Article < 4 Sentences	10	4953	4195
Article < 40 Tokens **Summary < 10 Tokens**	374	5446	10
Compression < 50%	10	1641	52802
Compression > 80%	11920	46776	456
Abstractivity < 10	0	6683	5942
Abstractivity > 80	227	303	42
Human-Eval(TeSum)	-	-	7183
Final Valid	**338**	**4006**	**20329**
Valid %	**2.6%**	**3.36%**	**21.9%**
Avg Abstractivity scores	68.23	36.44	31.08
Avg Compression(%)	71.71%	73.34%	58.24%

Table 4.1: Pre-processing and Filtration Details. Here, the bottom 2 rows show the average abstractivity scores and average compression% for the final valid pairs of the 3 datasets.

4.2.2 Automatic Quality Control

- *Compression ranges:* If an article is compressed [87] too much then we loose significant amount of information from the article, which contradicts the first property of summarization that all/most of the relevant information of article must be present in the summary. Though, a summary should also result in a significant amount of reduction in the size of the article but not at the cost of relevance. Therefore, we set compression% limits to be between 50-80. While the upper limit is higher than many previous datasets, (which,

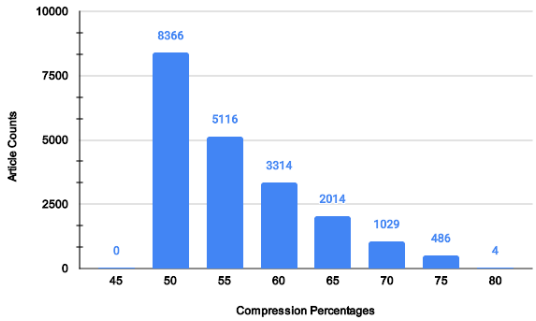

Figure 4.1: Article Count Vs. Compression

usually, set this to 30%) we find, particularly in news domain which is information dense, there can be large number of examples where slightly more content is required in the summary. Figure 4.1 shows the article counts of TeSum for compression% ranges.

- *Abstractivity ranges:* We want novelty in the summary, the content should be different from the source on both sentential as well as phrasal levels. We often find that even with the best editorial practices, the content in the highlights is often a conjunction of multiple disjoint phrases or absolute copy of phrases from the article. Which apart from being non-coherent, beats the third property of creativity. Therefore, we take the measure of abstractivity from [87] and apply 10-80 range of filtration. Even though we want the summary to be abstractive, we still need to copy some n-grams from the article which corresponds to factual information (names etc.) as presented in the article. Therefore, we restrict Abstractivity at 80 which is still a fairly lenient limit.

  On the lower side, for shorter articles which are information dense, it is possible that the copied unigrams or bigrams will constitute a large chunk of the summary in order to retain facts. This is especially true in the news domain where the summary creator has to copy smaller n-grams but would change the phrasal structure for novelty (paraphrasing). If abstractivity as proposed by Bommasani and Cardie [87] goes very close to zero then we get very high degree of copying in higher n-grams also. And on the other side, if the abstractivity is very high then we loose important information in terms of verbs, nouns, etc also which need to be there.

At this stage, after filtering by compression and abstractivity, we are left with 27512 article-summary pairs. Table 4.1 shows the number of article-summary pairs getting affected by each filter.

4.3 Human Evaluation

To maintain quality, one has to ensure that the human summarization guidelines are well understood by the creators and creators are, by an large, sticking to the guidelines. Though, it is impossible in any such task to have all the submissions manually evaluated, if a reasonable percentage of all the submissions are evaluated and found to be of high quality, it can be safely assumed that the rest of the submissions are also of high quality. Over the course of large number of evaluations, the expected percentage of lower quality samples in the total data can be estimated.

For human evaluation, the raters were asked to rate a minimum of 25% of the pairs from each HIT, for the 3 parameters *Relevance*, *Readability* and *Creativity* as per the Table 4.2. Each rater is supposed to rate a sample by giving scores, ranging between 0 to 4, for each parameter.

4.3.1 Special Cases in Human Evaluation

- If all the sentences are copied verbatim from the original article, scores are [0 0 0] for Relevance, Readability, and Creativity.
- In case of syntactic errors (spelling, spacing, punctuation), if that particular word/phrase deviates the overall meaning/context significantly, then scores will be deducted in Readability as well as Relevance.
- In case of tense issues, simultaneously, the scores can be reduced in Creativity and Relevance.
- The addition of irrelevant information or outside the context of the article leads to obtaining less scores in Creativity and Relevance.
- For anaphora-related issues, both Readability and Creativity scores will be reduced.
- Improper usage of novel words/phrases causes a reduction in Creativity score. If that particular word/phrase deviates from the original article's meaning, there will also be a reduction in the Relevance score.

4.3.2 Inter Rater Reliability

The inter-rater-reliability was established by following the guidelines (as mentioned in section 4.3). We ramdomly extracted 500 samples from the total collected articles. These 500 samples were then rated by 3 expert raters to compute the ICC3 scores. The agreement scores were then computed using the Intra-class Correlation Coefficient (ICC) [88] following the guidelines given by Koo and Li [89]. We report **ICC3** scores, which correspond to fixed raters and

	Relevance	**Readability**	**Creativity**
Score 0	No relevant information is covered or the entire summary is irrelevant	Summary is not at all understandable	Summary consists entirely of sentences copied verbatim from the 'original' article
Score 1	Only one relevant piece of information is covered or most of the summary is irrelevant	Most of the summary contains un-natural sentence structure and frequent grammatical errors	Copied most of the sentences from the 'original article' and contains rare novel words.
Score 2	Half of the relevant information is present or half of the irrelevant information is covered	Approximately half of the summary contains un-natural sentential/phrasal structures, which make the summary difficult to understand	Approximately half of the summary contains novel sentence structures and the remaining half is copied from the original article or few sentences are generated but have inaccurate meaning
Score 3	One piece of relevant information is missing, or one piece of irrelevant information is added	Summary is understandable but contains some errors in grammar, punctuation or spelling mistakes	Most of the summary is novel, but some of the non-factual content is copied verbatim (copied as it is)
Score 4	Everything is relevant and all the relevant information is covered	Summary is understandable and free from grammatical, punctuation and spelling mistakes	The entire summary, except the factual information (names, dates etc.), is novel

Table 4.2: Abstractive Summarization Evaluation Criteria

	Avg Scores			# Samples >= 3		
	XL-Sum	**MassiveSumm[Te]**	**TeSum**	**XL-Sum**	**MassiveSumm[Te]**	**TeSum**
Relevance	1	2	3	4	43	185
Readability	3.5	2.9	3.27	176	144	188
Creativity	0.98	1.58	3.28	12	51	170
All 3 parameters rated >= 3				**4**	**35**	**154**

Table 4.3: Human Evaluation of XL-Sum[Te], MassiveSumm[Te] and TeSum on 200 samples each.

individual (single) reliability. We specifically chose this model (ICC3), because each sampled article-summary pair, from the HITs, is then evaluated by one rater, and not all 3.

For our three parameters: Relevance, Readability and Creativity, our raters achieved **0.89**, **0.94** and **0.90** reliability scores respectively. These scores indicate good to excellent reliability.

4.3.3 Human Evaluation Process

Each HIT was evaluated by one rater, by randomly selecting a minimum of 25% from the HIT and distributing among the 3 raters, such that each pair of this 25% was evaluated by a single rater. If on an average the combination of these 25% pairs do not rate 3 or above for each individual parameter, then the entire HIT is rejected based on the assumption that there is a higher percentage of low quality submissions in this HIT. This process resulted in a total reduction of 7183 pairs. Giving us the final 20329 pairs.

Since we are evaluating only a percentage of the samples submitted in each HIT, we need to be aware of the possibility of some errors in the final dataset. To estimate this, we take the 5089 evaluated samples (25% of the final 20329) and find individual samples which have lower scores. These were found to be 3.6%. As, 25% is a fair enough sample size, we can safely extend the same error percentages to the entire dataset. Therefore resulting in a dataset which, while being smaller than other existing datasets, is of high quality. But if we subject the existing datasets to the same high standards that we expect from our dataset, we find that our dataset size is not low at all, in comparison with their resultant dataset sizes. The statistics of the TeSum dataset are presented in Table 4.4. This Table 4.4 contains information about avg unique words, sentences in article and summary for train, validation and test data.

	Train		Validation		Test	
# Pairs	16295		2017		2017	
Avg Compression%	58.26		58.08		58.28	
	Text	Summary	Text	Summary	Text	Summary
# Unique Words	183641	113723	49038	28873	49620	28777
Avg Unique Words	88.93	42.78	89.19	43.14	90.74	43.81
(Min, Max) Words	(30, 536)	(12, 213)	(32, 685)	(10, 248)	(36, 592)	(12, 261)
Avg Words	120.8	50.02	122.56	50.8	124.82	51.69
Avg Sentences	9.23	3.22	9.50	3.19	9.51	3.17

Table 4.4: TeSum Statistics

4.4 Evaluating Existing Datasets

For comparison, when we applied the filters mentioned in Section 4.2, to the existing datasets MassiveSumm and XL-Sum, we found some surprising results. As mentioned in the Table 4.1, MassiveSumm ended up with only 3.36% of their original dataset size. Similarly, XL-Sum also reduced to only 2.6% of their original size. Even if we relax the constraints a little bit, it does not help their end results much. We will be releasing all the filtering scripts along with the lists of IDs/URLs for basic problem cases from both the datasets.

4.4.1 Human Evaluation of MassiveSumm and XL-Sum

Due to surprising final numbers of the XL-Sum and MassiveSumm datasets after the filtration, we decided to validate this finding by manually evaluating randomly selected 200 article-summary pairs from each of the 3 datasets using the same raters. All samples were completely annonymized/randomized in order to avoid dataset bias. We found that the summaries are of low quality for XL-Sum and MassiveSumm on almost all parameters, Table 4.3 shows the average numbers obtained by each dataset for each parameter individually. Also, the counts of article-summary pairs from each dataset which gained 3 or above ratings are shown. The bottom line shows final count of valid pairs (out of the 200) for each dataset which were rated 3 or above, for all the 3 parameters.

	HINDI		MARATHI		GUJARATI	
	XL-Sum	**MassiveSumm**	**XL-Sum**	**MassiveSumm**	**XL-Sum**	**MassiveSumm**
Dataset Size	**88472**	**563477**	**13627**	**127838**	**11397**	**43830**
Empty	5	20936	1	1488	0	3797
Duplicate Pairs	4	48461	0	614	1	525
Duplicate Summary	698	5626	465	4507	59	878
Prefixes	19	4225	3	4015	6	99
Article < 4 Sentences	164	27845	6	6811	104	5307
Article < 40 Tokens **Summary < 10 Tokens**	377	125372	154	60489	97	14303
Compression < 50%	13	1990	6	145	5	163
Compression > 80%	85028	286696	11985	47659	10611	18481
Abstractivity < 10	4	10668	1	843	0	55
Abstractivity > 80	29	643	411	128	91	11
Final Valid	**2131**	**31015**	**595**	**1139**	**423**	**211**
Valid %	**2.4%**	**5.5%**	**4.37%**	**0.89%**	**3.71%**	**0.48%**

Table 4.5: Filtration counts of XL-Sum and MassiveSumm for the other 3 languages; Hindi, Marathi and Gujarati.

4.4.2 Human Evaluation of other language datasets

As the numbers on the existing datasets were too surprising, we wondered if it was for this particular language. Therefore, we extended our analysis to some other languages (Hindi, Gujarati and Marathi from both XL-Sum and MassiveSumm) that we could evaluate (could read). We found similar issues in all of these datasets. We show the detailed filtration counts in Table 4.5.

Showed examples of some of the common problem cases (from the respective datasets). Instead of Telugu script, we have used phonetic transcription of Telugu using ISO15919, which is similar to IAST, for better readability. Examples are showed in Table 4.6, 4.7, 4.8, 4.9, and 4.10.

4.5 Conclusion

We constructed a high quality Human-curated Abstractive summarization dataset for Telugu. We also compared the dataset properties with existing Telugu summarization datasets and

Corpus	XL-Sum
IDs	international-54722433 international-55923039
URLs	https://www.bbc.com/telugu/international-54722433 https://www.bbc.com/telugu/international-55923039
Text	Article-1: lākḍaun valla cālāmaṁdi iḷlakē parimitaṁ ayyāru. ī yuvati khāḷīgā kūrcōkuṁḍā mēkap braṣ paṭṭukuni pramukhullā tayārai prācuryaṁ poṁdāru. (bībīsī telugunu phēsbuk , insṭāgrām , ṭviṭarlō phālō avvaṁḍi. yūṭyūblō sabskraib cēyaṁḍi.) Article-2: 2013 lō vēls rājadhāni kārḍiph nuṁci siriyā veḷli aisislō cērāru koṁdaru ṭīnējarlu. vāḷlu aisislō cēraḍāniki kāraṇālēṁṭō telusukōvālani bībīsī pratinidhi olīviyā vārini iṁṭarvyū cēśāru. iṁtakī dēśālu dāṭi veḷli aisislō cērina vāḷlaṁtā akkaḍa telusukunna vāstavālēṁṭi ? (bībīsī telugunu phēsbuk , insṭāgrām, ṭviṭarlō phālō avvaṁḍi. yūṭyūblō sabskraib cēyaṁḍi.)
Summary	ivi kūḍā cadavaṁḍi:
Remark	17 different articles have the same summary

Table 4.6: XL-Sum: Duplicate Summary example

Corpus	MassiveSumm
URLs	https://telugu.asianetnews.com/astrology/today-may-1st-2019-your-horoscope-pqt03m https://telugu.asianetnews.com/astrology/today-2nd-july-2019-tuesday-your-horoscope-ptzqhq
Text	Article-1: mēṣaṁ : (aśvini , bharaṇi , kṛttika 1 vapādaṁ) peddalaṁṭē gauravaṁ perugutuṁdi . ādhyātmika ciṁtana perugutuṁdi śāstra parijñānaṁ pai dṛṣṭi ērpaḍutuṁdi . viśāla bhāvālu uṁṭāyi . vidya nērcukōvaḍaṁ valla vaccē gauravaṁ perugutuṁdi . . . .rājakīyālapai dṛṣṭi sāristāru. gauravaṁ peṁcukunē prayatnaṁ. vṛtti udyōgādullō ottiḍulu uṁṭāyi . śrī mātrē namaḥ japaṁ maṁcidi. Article-2: mēṣaṁ : (aśvini , bharaṇi , kṛttika 1 vapādaṁ) racanalapai dṛṣṭi taggutuṁdi . kamyūnikēṣans valla anukūlata perugutuṁdi . parāmarśalu cēstāru . pracārālapai dṛṣṭi ērpaḍutuṁdi . baṁdhuvula sahakāraṁ labhistuṁdi . prayāṇāla valla jāgratta avasaraṁ…….vidyārthulaku kaṭhinamena samayaṁ . ālōcanallō ottiḍi ērpaḍutuṁdi . durgādēvi pūja cēsukōvaḍaṁ śubha phalitālanistuṁdi
Summary	ī rōju rāśiphalālu ilā unnāyi
Remark	Many articles (with different URLs) have the same summary

Table 4.7: MassiveSumm: Duplicate summary example

Corpus	MassiveSumm
URL	https://telugu.asianetnews.com/entertainment-news/sridevi-s-second-death-anniversary-prayer-meet-in-chennai-q6oh3x
Text	2018 phibravari 24 na dubāy lō śrīdēvi anumānāspada sthitilō maraṇiṁcāru. atilōka suṁdarigā śrīdēvi iṁḍiyā mottaṁ tirugulēni krēj soṁtaṁ cēsukuṁdi. śrīdēvi akāla maraṇaṁ ceṁdaḍaṁtō citra pariśrama tōpāṭu abhimānulu kūḍā tīvra viṣādāniki gurayyāru gata phibravari 2 4 na ku śrīdēvi maraṇiṁci reṁḍēḷlu pūrtayiṁdi. . …….amma nuvvu ikkaḍē uṁḍālani kōrukuṁṭunnā ani jānvī kāmeṁṭ peṭṭiṁdi . jānvī sṭār hīrōyin gā rāṇiṁcālanēdi śrīdēvi kala . prastutaṁ jānvī bālīvuḍ lō palu citrāllō naṭistōṁdi .
Summary	2018 phibravari 24 na dubāy lō śrīdēvi anumānāspada sthitilō maraṇiṁcāru. atilōka suṁdarigā śrīdēvi iṁḍiyā mottaṁ tirugulēni krēj soṁtaṁ cēsukuṁdi
Remark	The highlighted content is the prefix information

Table 4.8: MassiveSumm: Prefix example

Corpus	XL-Sum
Language	Telugu
ID	international-41926617
URL	https://www.bbc.com/telugu/international-41926617
Text	prāṇālanu guppiṭlō peṭṭukoni lakṣala maṁdi prajalu śaraṇārthulugā nagarānni vadaliveḷlāru. vēla maṁdi maraṇiṁcāru. eṁtō maṁdi kuṭuṁba sabhyulanu kōlpōyi tīvra vēdanaku guravutunnāru. alā sarvaṁ kōlpōyina ō bādhituḍi vyatha idi. ī vīḍiyōnu bībīsī arabik rūpoṁdiṁciṁdi. mā itara kathanālu : (bībīsī telugunu phēs buk, in sṭāgrām, ṭviṭar lō phālō avvaṁḍi. yūṭyūb lō sab skraib cēyaṁḍi.)
Summary	aies miliṭeṁṭlaku, sainika balagālaku madhya jarigina pōrulō siriyālōni rakhā nagaraṁ dhvaṁsamaiṁdi.iḷlannī dhvaṁsamayyāyi.

Table 4.9: XL-Sum(Telugu): Out of the context example

Corpus	XL-Sum
Language	Marathi
ID	media-54453803
URL	https://www.bbc.com/marathi/media-54453803
Text	up-rāṣṭrādhyakṣālā rāṣṭrādhyakṣācā raniṁg mēṭ mhaṇajēc sāthīdār mhaṭalaṁ jātaṁ . up - rāṣṭrādhyakṣācyā umēdavārācyā pratiṣṭhēvar āṇi kāryakṣamatēkaḍē pāhūnahī matadān karaṇārā varg amērikēt āhē . aśāvēḷī ḍēmōkrēṭik umēdavār kamalā hêris yāṁcyāviṣayī tumhālā adhik jāṇūn ghyāyacaṁ āhē . pāhā hā vhīḍiō . . hēhī pāhilaṁt kā ? (bībīsī marāṭhīcē sarv apaḍēṭs miḷavaṇyāsāṭhī tumhī āmhālā phēsabuk ,insṭāgrām , yūṭyūb , ṭviṭar var phôlō karū śakatā . ' bībīsī viśv ' rōj samdhyākāḷī 7 vājatā JioTV êp āṇi yūṭyūbavar nakkī pāhā .)
Summary	amērikētē prēsiḍēṁśiēl āṇi vhāīs prēsiḍēṁśiēl ḍibēṭalā mahattv asataṁ. yā dōnhī ḍibēṭamuḷē sarakāracī diśā āṇi dhyēyadhōraṇaṁ lōkāṁnā samajatāt .

Table 4.10: XL-Sum(Marathi): Out of the context example

claim that these existing datasets can also benefit from the quality control measures that we have proposed. Though, purely on the basis of size, our work also started with a huge collection of 92k+ article-summary pairs like the existing datasets, but by making use of human expertise at both annotation and quality assessment stages, we show that after applying the same quality measures our dataset performs significantly better then the automated ones. And as a result we out-perform the other datasets in terms of final size as well.

Chapter 5

Baselines for Telugu abstractive summarization

We have constructed a high quality Human-curated Abstractive summarization dataset for Telugu (TeSum). In this chapter we have presented different summarization architectures trained on this summarization dataset. The input to the Summarization pipeline is vector representation of the article and out is summary of the article. To perform the automatic summarization task, we have implemented the sequence-to-sequence [80] Recurrent Neural Networks (RNN) model with attention mechanism [90] and pointer-generator [79] with coverage mechanism. Further, we used the novel intra-attention mechanism [81] with reinforcement learning (RL). A novel document-level encoder [91] using Bidirectional Encoder Representations from Transformers (BERT) can be used for both extractive and abstractive summarization. We fine-tuned the multilingual text to text transfer transformer (mT5) [92] with TeSum corpus. We have elaborated a detailed explanation of the benchmark models in below section.

5.1 Baseline Models

We have presented some common baselines used for summarization by other authors, to demonstrate the impact of the datasets on summarization using various models. In order to show the proof of quality of TeSum dataset on the summarization task itself, and to provide various baselines, we trained and tested several existing summarization models with TeSum data. Below we have given brief introduction of different architectures.

1. **Pointer-Generator:** This model is implemented using [80] sequence-to-sequence Recurrent Neural Networks (RNN) with [90] attention mechanism. Furthermore, we have used the pointer-generator[79] and coverage mechanism in the model. Pointer-generator will help in deciding whether to copy words from the source text or generate from the vocabulary. Hence pointer mechanism effectively handles the Out Of Vocabulary (OOV) words problem. The coverage mechanism, prevents the model not attending to the same phrases multiple times to handle the repetition problem in summaries.

2. **ML+RL with intra-attention:** This model implemented the novel intra-attention mechanism [81] that attends over the input document and continuously generates decoder output separately to reduce the problem of repetitive and incoherent phrases in the summaries. Furthermore, they introduced a new training method that combined supervised and Reinforcement Learning (RL). That will prevent exposure bias problems and can produce readable summaries.

3. **Text Summarization with Pretrained Encoders:** According to this model, introduced a novel document-level encoder [91] using Bidirectional Encoder Representations from Transformers (BERT) that can be used for both extractive and abstractive summarization. The extractive model was built by stacking many inter-sentence Transformers layers on top of this novel encoder. For abstractive summarization, this method adopted the encoder-decoder architecture with a new fine-tuning approach where the encoder is pre-trained BERT and the decoder is randomly initialized Transformer. It also explored with a two-stage approach, where they first trained with an extractive object and then further trained with an abstractive objective. We have used the embeddings [93] trained on 8M+ Telugu sentences to do the experiments with the BERTSum model.

4. **mT5:** We finetuned the multilingual text to text transfer transformer (mT5) [92] on TeSum corpus and we used mT5-base for our experiments. The mT5 model is the variant of the T5 [94] model trained on common crawl dataset.

5.2 Experimental Setup

In this section, we have explained the dataset statistics for Tesum, MassiveSum, and XL-Sum. We have presented experimental details of different architectures.

5.2.1 TeSum Dataset

To create train, dev and test splits of TeSum dataset, we divide the total 20329 pairs into carefully selected sub-parts of about 80%, 10% and 10% respectively. The selection of pairs is done in a way that preserves the balance in terms of length of articles, compression(%) and abstractivity levels across the three splits. In chapter 4, Table 4.4 details the statistics for the 3 splits.

5.2.2 MassiveSum and XL-Sum

As, after our filtration steps, the originally large-scale existing-datasets ended up with a very low percentage of their total article-summary pairs. Which extrinsically does not make for a

Parameters	PG+	MLE+	BertSumAbs	mT5
Max source length	400	400	512	512
Max target length	100	100	200	256
Min target length	35	35	50	30
Batch Size	8	8	140	2
Epochs/Iterations	100k iter	100k iter	50k iter	10 epochs
Vocab Size	50k	50k	28996	250112
Beam Size	4	4	5	4
Learning Rate	0.15	0.001 (MLE) 0.0001 Others	lr_bert = 0.002 lr_dec = 0.2	5e-4

Table 5.1: Experimental setup and parameter settings

fair comparison. Therefore, before going ahead with the model training and experiments, for evaluating the effect of these curations of the datasets for the task of summarization, we are forced to make some concessions for XL-Sum and MassiveSumm.

As a concession for MassiveSumm, we decided to concede compression from 80% to 90% and we find that it added a fairly high number of articles to the valid set for MassiveSumm (giving us a total of 17248 pairs, which we then divide into about 80%-10%-10% to get the train, dev and test splits). Relaxing the compression further would increase the numbers, but we also note that the authors themselves have presented their results on a randomly selected 12633 pairs (not made available by the author), therefore we take a comparative number, which according to us should be of a better quality due to the aggressive quality control.

For XL-Sum, the only option was to remove all constraints, as the original size itself was quite small. Therefore, we considered the original splits of XL-Sum (Telugu) for our experiments.

5.2.3 Parameters of different summarization architectures

For the experiments and baseline training, we have used Word2Vec [95] (Telugu Wikipedia pre-trained) embeddings. Apart from mT5, which was fine-tunned using 2 GPUs and 20 CPUs, the rest had system config of 1 GPU and 10 CPUs. Further details on hyper-parameter settings and configuration is listed in Table 5.1. Here, 'PG+' represents PG and PG+Coverage models, and 'MLE+' represents MLE, MLE+RL and RL models.

5.3 Results & Analysis

For better comparison, experiments were conducted by training on each dataset's training split and then testing on all 3 datasets' test set. Table 5.2 shows the ROUGE scores[1] for some of the selected best performing model configurations. Here, 'wo' with MLE+RL and RL models stands for 'without intra attention', and 'Pointer Generator' represents the PG+Coverage model.

Looking at this table our first observation is that models trained on TeSum end up performing well across the board, but do not end up beating models trained and tested on the same dataset for almost all models. We surmise that this is because the fundamental nature of these summarization datasets is different. While MassiveSumm and XL-Sum summaries are primarily small number of disjoint sentences, TeSum summaries are coherent discourses in themselves. This means that a model trained to avoid copying and trained to generate coherent discourse would fail on MassiveSumm and XL-Sum.

5.4 Conclusion

In this chapter, we have evaluated the performance of summarization architectures on different datasets like TeSum, MassiveSum, and XL-Sum. While we accept the contributions made by XL-Sum and MassiveSumm, which bring value to this field for any given language, we claim that this scraping and the initial pre-processing is just the first step. The data need to be held to higher standards. Even if it is achieved by scraping, filtering and then evaluating a percentage of randomly selected samples of the resultant, it would ensure a much more valuable dataset than just scraping.

[1] Multilingual Rouge from XL-Sum https://tinyurl.com/MLTERouge

Trained on TeSum

	Tested on TeSum			Tested on MassiveSumm			Tested on XL-Sum		
Model	**T-R1**	**T-R2**	**T-RL**	**M-R1**	**M-R2**	**M-RL**	**XL-R1**	**XL-R2**	**XL-RL**
Pointer Generator	**39.37**	**22.72**	**32.15**	25	**13.8**	20.74	**9.73**	**2.29**	**7.32**
MLE + RL- wo	38.09	21.9	31.77	**25.05**	13.41	**20.78**	8.56	2.03	6.54
RL- wo	31.19	17.6	24.86	20.16	10.58	16.75	8.28	1.96	6.45
BertSumAbs	26.49	12.55	19.6	18.61	8.24	14.69	6.21	1.34	4.96
mT5-small	37.42	20.82	30.88	24.37	12.5	20.2	8.8	2.06	6.7

Trained on MassiveSumm

	Tested on TeSum			Tested on MassiveSumm			Tested on XL-Sum		
Model	**T-R1**	**T-R2**	**T-RL**	**M-R1**	**M-R2**	**M-RL**	**XL-R1**	**XL-R2**	**XL-RL**
Pointer Generator	26.31	14.45	22.16	28.38	16.33	24.95	**9.85**	**2.18**	**8.34**
MLE + RL- wo	26.3	14.62	22.36	**30.46**	**18.69**	**27.17**	9.82	2.03	8.23
RL- wo	15.59	7.93	13.75	13.82	7.76	12.82	4.27	0.89	3.88
BertSumAbs	**29.73**	13.59	22.02	23.76	11.47	19.28	7.69	1.54	6.11
mT5-small	27.67	**15.08**	**23.03**	29.43	17.41	26.25	9.67	1.91	8.14

Trained on XL-Sum

	Tested on TeSum			Tested on MassiveSumm			Tested on XL-Sum		
Model	**T-R1**	**T-R2**	**T-RL**	**M-R1**	**M-R2**	**M-RL**	**XL-R1**	**XL-R2**	**XL-RL**
Pointer Generator	17.13	2.2	10.06	12.45	1.59	8.73	5.41	0.28	4.35
MLE + RL- wo	3.7	0.44	3.18	2.88	0.43	2.63	1.17	0.04	1.13
RL- wo	2.4	0.28	2.14	1.61	0.17	1.49	0.68	0.04	0.66
BertSumAbs	13.8	2.41	10.26	11.46	2.06	9.19	6.55	1.44	5.73
mT5-small	**18.42**	**9**	**15.88**	**19.44**	**9.12**	**17.46**	**12.24**	**3.6**	**11.18**

Table 5.2: ROUGE scores achieved by various baseline models.

Chapter 6

Conclusions

In this thesis, we have explored and addressed the challenges involved in the creation of language resources and systems for a low resource language, Telugu. The main contributions of this thesis include both, resource creation and system development. At resource creation level, we have manually created a question classification database and a human-generated abstractive summarization corpus for Telugu. At system development level, we have implemented *AVADHAN* an end to end architecture for open domain question-answering in Telugu; and also some end-to-end neural architecture baselines for abstractive Telugu summarization.

The following summarises the conclusions of this thesis.

- We have manually annotated question classification data which consists of 1037 pairs. It contains seven categories of questions.
- We have explored different classifiers for question classification with manually annotated question-answer pairs. Implemented end to end pipeline for question-answering.
- We have proposed novel guidelines for creation and evaluation of summarization dataset which can be useful for creating high quality abstractive summaries.
- We have performed aggressive evaluation to create high-quality summaries and also performed existing dataset quality assessment.
- Our corpus TeSum consists of 20329 summary pairs which is high quality large abstractive summarization database available in Telugu.
- We have presented baseline abstractive summarization results for TeSum dataset.

6.1 Future Scope

Some directions for future scope are as follows:

- The end to end pipeline AVADHAN can be extended to other languages by choosing the language specific question classification module.
- The proposed novel guidelines for the creation of summaries can be extend to other languages.
- The proposed quality assessment process can be applied for scraped dataset to maintain high-quality that deserve the abstractive summarization properties.
- Created summarization dataset can be further extend to downstream tasks like headline generation, question answering dataset.

Related Publications

Conference

1. Priyanka Ravva, Ashok Urlana, and Manish Shrivastava. "AVADHAN: System for Open-Domain Telugu Question Answering." Proceedings of the 7th ACM IKDD CoDS and 25th COMAD. 2020. 234-238

2. Ashok Urlana, Nirmal Surange, Pavan Baswani, Priyanka Ravva and Manish Shrivastava. "TeSum:Human-Generated Abstractive Summarization Corpus for Telugu." Proceedings of the 13th LREC 20 to 25 June 2022. 5712–5722.

Bibliography

[1] D. Sengupta and G. Saha, "Study on similarity among indian languages using language verification framework," *Advances in Artificial Intelligence*, vol. 2015, pp. 2–2, 2015. 1

[2] A. Kunchukuttan, D. Kakwani, S. Golla, A. Bhattacharyya, M. M. Khapra, P. Kumar, *et al.*, "Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages," *arXiv preprint arXiv:2005.00085*, 2020. 2

[3] B. Jawaid, A. Kamran, and O. Bojar, "A tagged corpus and a tagger for urdu.," in *LREC*, vol. 2, pp. 2938–2943, 2014. 2

[4] G. Ramesh, S. Doddapaneni, A. Bheemaraj, M. Jobanputra, R. AK, A. Sharma, S. Sahoo, H. Diddee, D. Kakwani, N. Kumar, *et al.*, "Samanantar: The largest publicly available parallel corpora collection for 11 indic languages," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 145–162, 2022. 2

[5] D. Kanojia, K. Patel, and P. Bhattacharyya, "Indian language wordnets and their linkages with princeton wordnet," *arXiv preprint arXiv:2201.02977*, 2022. 2

[6] X. Pan, B. Zhang, J. May, J. Nothman, K. Knight, and H. Ji, "Cross-lingual name tagging and linking for 282 languages," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1946–1958, 2017. 2

[7] Y. Madhani, S. Parthan, P. Bedekar, R. Khapra, V. Seshadri, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Aksharantar: Towards building open transliteration tools for the next billion users," *arXiv preprint arXiv:2205.03018*, 2022. 2

[8] M. M. Yoonus and S. Sinha, "A hybrid pos tagger for indian languages.," *Language in India*, vol. 11, no. 9, 2011. 2

[9] H. Agrawal, "Pos tagging and chunking for indian languages," *Shallow parsing for South Asian languages*, p. 37, 2007. 2

[10] P. Kosaraju, S. R. Kesidi, V. B. R. Ainavolu, and P. Kukkadapu, "Experiments on indian language dependency parsing," *Proceedings of the ICON10 NLP Tools Contest: Indian Language Dependency Parsing*, pp. 40–45, 2010. 2

[11] D. Kakwani, A. Kunchukuttan, S. Golla, N. Gokul, A. Bhattacharyya, M. M. Khapra, and P. Kumar, "Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4948–4961, 2020. 2

[12] A. Kumar, H. Shrotriya, P. Sahu, A. Mishra, R. Dabre, R. Puduppully, A. Kunchukuttan, M. M. Khapra, and P. Kumar, "Indicnlg benchmark: Multilingual datasets for diverse nlg tasks in indic languages," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5363–5394, 2022. 2

[13] J. H. Clark, E. Choi, M. Collins, D. Garrette, T. Kwiatkowski, V. Nikolaev, and J. Palomaki, "Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 454–470, 2020. 2

[14] J. Liu, Y. Lin, Z. Liu, and M. Sun, "Xqa: A cross-lingual open-domain question answering dataset," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2358–2368, 2019. 2

[15] D. Gupta, S. Kumari, A. Ekbal, and P. Bhattacharyya, "Mmqa: A multi-domain multilingual question-answering framework for english and hindi," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018. 2

[16] T. Hasan, A. Bhattacharjee, M. S. Islam, K. Mubasshir, Y.-F. Li, Y.-B. Kang, M. S. Rahman, and R. Shahriyar, "XL-sum: Large-scale multilingual abstractive summarization for 44 languages," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, (Online), pp. 4693–4703, Association for Computational Linguistics, Aug. 2021. 2, 3, 5, 10, 26

[17] D. Varab and N. Schluter, "Massivesumm: a very large-scale, very multilingual, news summarisation dataset," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10150–10161, 2021. 2, 3, 26

[18] M. H. Khanam and S. Sravani, "Text summarization for telugu document," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 18, no. 6, pp. 25–28, 2016. 3, 9

[19] K. K. Mamidala *et al.*, "A heuristic approach for telugu text summarization with improved sentence ranking," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 3, pp. 4238–4243, 2021. 3

[20] R. Naidu, S. K. Bharti, K. S. Babu, and R. K. Mohapatra, "Text summarization with automatic keyword extraction in telugu e-newspapers," in *Smart Computing and Informatics*, pp. 555–564, Springer, 2018. 3, 10

[21] J. S. Kallimani, K. Srinivasa, *et al.*, "Information retrieval by text summarization for an indian regional language," in *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering (NLPKE-2010)*, pp. 1–4, IEEE, 2010. 3, 10

[22] S. Shashikanth and S. Sanghavi, "Text summarization techniques survey on telugu and foreign languages," *International Journal of Research in Engineering, Science and Management*, vol. 2, no. 1, 2019. 3

[23] D. N. S. Y Madhavee Latha, "Multi-document abstractive text summarization through semantic similarity matrix for telugu language," *International Journal of Advanced Science and Technology*, vol. 29(1), pp. 513–521, 2020. 3, 10

[24] J. S. Kallimani, K. Srinivasa, and B. E. Reddy, "Statistical and analytical study of guided abstractive text summarization," *Current Science*, pp. 69–72, 2016. 3

[25] B. M. Bharath, B. A. Gowtham, and M. Akhil, "Neural abstractive text summarizer for telugu language," in *Soft Computing and Signal Processing*, pp. 61–70, Springer, 2022. 3, 5

[26] J. K. Geetha and N. Deepamala, "Kannada text summarization using latent semantic analysis," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1508–1512, IEEE, 2015. 4

[27] R. Jayashree, K. Srikanta, and K. Sunny, "Document summarization in kannada using keyword extraction," *Proceedings of AIAA*, vol. 11, pp. 121–127, 2011. 4

[28] R. Balabantaray, B. Sahoo, D. Sahoo, and M. Swain, "Odia text summarization using stemmer," *Int. J. Appl. Inf. Syst*, vol. 1, no. 3, pp. 2249–0868, 2012. 4

[29] A. Burney, B. Sami, N. Mahmood, Z. Abbas, and K. Rizwan, "Urdu text summarizer using sentence weight algorithm for word processors," *International Journal of Computer Applications*, vol. 46, no. 19, pp. 38–43, 2012. 4, 10

[30] M. Gupta and N. K. Garg, "Text summarization of hindi documents using rule based approach," in *2016 international conference on micro-electronics and telecommunication engineering (ICMETE)*, pp. 366–370, IEEE, 2016. 4

[31] K. Sarkar, "Bengali text summarization by sentence extraction," *arXiv preprint arXiv:1201.2240*, 2012. 4, 10

[32] P. Krishnaprasad, A. Sooryanarayanan, and A. Ramanujan, "Malayalam text summarization: An extractive approach," in *2016 International Conference on Next Generation Intelligent Systems (ICNGIS)*, pp. 1–4, IEEE, 2016. 4

[33] V. Gupta and G. S. Lehal, "Automatic punjabi text extractive summarization system," in *Proceedings of COLING 2012: Demonstration Papers*, pp. 191–198, 2012. 4

[34] V. V. Sarwadnya and S. S. Sonawane, "Marathi extractive text summarizer using graph based model," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–6, IEEE, 2018. 4

[35] D. Sindhu and B. Sagar, "Dictionary based machine translation from kannada to telugu," in *IOP conference series: materials science and engineering*, vol. 225, p. 012182, IOP Publishing, 2017. 3

[36] K. Lingam, E. R. Lakshmi, and L. R. Theja, "Rule-based machine translation from english to telugu with emphasis on prepositions," in *2014 First International Conference on Networks & Soft Computing (ICNSC2014)*, pp. 183–187, IEEE, 2014. 3

[37] A. Hegde, S. Banerjee, B. R. Chakravarthi, R. Priyadharshini, H. Shashirekha, J. P. McCrae, *et al.*, "Overview of the shared task on machine translation in dravidian languages," in *Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages*, pp. 271–278, 2022. 3

[38] S. S. Mukku, N. Choudhary, and R. Mamidi, "Enhanced sentiment classification of telugu text using ml techniques," in *SAAIP@ IJCAI*, 2016. 3

[39] R. Naidu, S. K. Bharti, K. S. Babu, and R. K. Mohapatra, "Sentiment analysis using telugu sentiwordnet," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 666–670, IEEE, 2017. 3

[40] S. Tammina, "A hybrid learning approach for sentiment classification in telugu language," in *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, pp. 1–6, IEEE, 2020. 3

[41] G. Raju, S. Badugu, and V. Akhila, "Telugu text classification using supervised machine learning algorithm," in *Smart Intelligent Computing and Applications, Volume 1*, pp. 293–305, Springer, 2022. 3

[42] V. V. R. Vegesna, K. Gurugubelli, H. K. Vydana, B. Pulugandla, M. Shrivastava, and A. K. Vuppala, "Dnn-hmm acoustic modeling for large vocabulary telugu speech recognition," in *International Conference on Mining Intelligence and Knowledge Exploration*, pp. 189–197, Springer, 2017. 3

[43] R. Venkateswarlu, R. R. Teja, and R. V. Kumari, "Developing efficient speech recognition system for telugu letter recognition," in *2012 International Conference on Computing, Communication and Applications*, pp. 1–6, IEEE, 2012. 3

[44] R. R. N. Reddy and S. Bandyopadhyay, "Dialogue based question answering system in telugu," in *Proceedings of the Workshop on Multilingual Question Answering*, pp. 53–60, Association for Computational Linguistics, 2006. 5, 9

[45] M. C. Sravanthi, K. Prathyusha, and R. Mamidi, "A dialogue system for telugu, a resource-poor language," in *International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 364–374, Springer, 2015. 5, 9

[46] P. Danda, P. Jwalapuram, and M. Shrivastava, "End to end dialog system for telugu," in *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pp. 265–272, 2017. 5, 9

[47] S. R. Duggenpudi, K. S. S. Varma, and R. Mamidi, "Samvaadhana: A telugu dialogue system in hospital domain," in *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pp. 234–242, 2019. 5, 9

[48] C. Yuan and C. Wang, "Parsing model for answer extraction in chinese question answering system," in *2005 International Conference on Natural Language Processing and Knowledge Engineering*, pp. 238–243, IEEE, 2005. 8

[49] M. E. Sucunuta and G. E. Riofrio, "Architecture of a question-answering system for a specific repository of documents," in *2010 2nd International Conference on Software Technology and Engineering*, vol. 2, pp. V2–12, IEEE, 2010. 8

[50] K. Ishwari, A. Aneeze, S. Sudheesan, H. Karunaratne, A. Nugaliyadde, and Y. Mallawarrachchi, "Advances in natural language question answering: A review," *arXiv preprint arXiv:1904.05276*, 2019. 8

[51] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996. 8

[52] A. I. Martin, M. Franz, and S. Roukos, "Ibm's statistical question answering system-trec-10," in *In Proceedings of the Tenth Text REtrieval Conference (TREC*, Citeseer, 2001. 8

[53] X. Li and D. Roth, "Learning question classifiers," in *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pp. 1–7, Association for Computational Linguistics, 2002. 8

[54] S. Hakimov, S. Jebbara, and P. Cimiano, "Deep learning approaches for question answering on knowledge bases: an evaluation of architectural design choices," *arXiv preprint arXiv:1812.02536*, 2018. 8

[55] D. Sorokin and I. Gurevych, "Modeling semantics with gated graph neural networks for knowledge base question answering," *arXiv preprint arXiv:1808.04126*, 2018. 8

[56] Z. Zheng, "Answerbus question answering system," in *Proceedings of the second international conference on Human Language Technology Research*, pp. 399–404, Morgan Kaufmann Publishers Inc., 2002. 8

[57] D. Zhang and W. S. Lee, "A web-based question answering system," 2003. 8

[58] S. Sahu, N. Vasnik, and D. Roy, "Prashnottar: a hindi question answering system," *International Journal of Computer Science & Information Technology*, vol. 4, no. 2, p. 149, 2012. 9

[59] M. Vargas-Vera, E. Motta, and J. Domingue, "Aqua: An ontology-driven question answering system.," *New Directions in Question Answering*, vol. 8, 2003. 9

[60] P. Ranjan and R. C. Balabantaray, "Question answering system for factoid based question," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, pp. 221–224, IEEE, 2016. 9

[61] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus, "The structure and performance of an open-domain question answering system," in *Proceedings of the 38th annual meeting on association for computational linguistics*, pp. 563–570, Association for Computational Linguistics, 2000. 9

[62] P. Gupta and V. Gupta, "Hybrid approach for punjabi question answering system," in *Advances in Signal Processing and Intelligent Recognition Systems*, pp. 133–149, Springer, 2014. 9

[63] S. Archana, N. Vahab, R. Thankappan, and C. Raseek, "A rule based question answering system in malayalam corpus using vibhakthi and pos tag analysis," *Procedia Technology*, vol. 24, pp. 1534–1541, 2016. 9

[64] I. Seena, G. Sini, and R. Binu, "Malayalam question answering system," *Procedia Technology*, vol. 24, pp. 1388–1392, 2016. 9

[65] G. Nanda, M. Dua, and K. Singla, "A hindi question answering system using machine learning approach," in *2016 international conference on computational techniques in information and communication technologies (ICCTICT)*, pp. 311–314, IEEE, 2016. 9

[66] S. Banerjee, S. K. Naskar, and S. Bandyopadhyay, "Bfqa: A bengali factoid question answering system," in *International Conference on Text, Speech, and Dialogue*, pp. 217–224, Springer, 2014. 9

[67] S. Sarker, S. T. A. Monisha, and M. M. H. Nahid, "Bengali question answering system for factoid questions: A statistical approach," in *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–5, IEEE, 2019. 9

[68] A. Das, J. Mandal, Z. Danial, A. Pal, and D. Saha, "A novel approach for automatic bengali question answering system using semantic similarity analysis," *International Journal of Speech Technology*, vol. 23, no. 4, pp. 873–884, 2020. 9

[69] M. R. Bhuiyan, A. K. M. Masum, M. Abdullahil-Oaphy, S. A. Hossain, and S. Abujar, "An approach for bengali automatic question answering system using attention mechanism," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–5, IEEE, 2020. 9

[70] R. Damodar, A. Ramineni, and R. Konda, "Telugu text summarization," *International Research Journal of Modernization in Engineering Technology and Science, India*, 2021. 9

[71] K. U. Manjari, "Extractive summarization of telugu documents using textrank algorithm," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 678–683, IEEE, 2020. 9

[72] P. Pingali, J. Jagarlamudi, and V. Varma, "A dictionary based approach with query expansion to cross language query based multi-document summarization: Experiments in telugu-english," *National Workshop on Artificial Intelligence*, 2006. 10

[73] S. Renjith and P. Sony, "An automatic text summarization for malayalam using sentence extraction," in *proceedings of 27th IRF International Conference*, 2015. 10

[74] S. Pattnaik and A. K. Nayak, "A simple and efficient text summarization model for odia text documents," *Indian journal of computer science and engineering*, vol. 11, 2020. 10

[75] C. Thaokar and L. Malik, "Test model for summarizing hindi text using extraction method," in *2013 IEEE Conference on Information & Communication Technologies*, pp. 1138–1143, IEEE, 2013. 10

[76] M. Hanumanthappa, M. Narayana Swamy, and N. Jyothi, "Automatic keyword extraction from dravidian language," *International Journal of Innovative Science Engineering and Technology*, vol. 1, no. 8, pp. 87–92, 2014. 10

[77] S. Bhosale, D. Joshi, V. Bhise, and R. Deshmukh, "Marathi e-newspaper text summarization using automatic keyword extraction technique," *International Journal of Advance Engineering and Research Development*, vol. 5, no. 3, pp. 789–792, 2018. 10

[78] J. S. Kallimani, K. Srinivasa, *et al.*, "Information extraction by an abstractive text summarization for an indian regional language," in *2011 7th International Conference on Natural Language Processing and Knowledge Engineering*, pp. 319–322, IEEE, 2011. 10

[79] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 1073–1083, Association for Computational Linguistics, July 2017. 10, 40

[80] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014. 10, 40

[81] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," *arXiv preprint arXiv:1705.04304*, 2017. 10, 40, 41

[82] Y.-C. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting," *arXiv preprint arXiv:1805.11080*, 2018. 10

[83] P. M. Shishtla, K. Gali, P. Pingali, and V. Varma, "Experiments in telugu ner: A conditional random field approach," in *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*, 2008. 21

[84] A. Ekbal, R. Haque, A. Das, V. Poka, and S. Bandyopadhyay, "Language independent named entity recognition in indian languages," in *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*, 2008. 21

[85] B. Sasidhar, P. Yohan, A. V. Babu, and A. Govardhan, "A survey on named entity recognition in indian languages with particular reference to telugu," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 2, p. 438, 2011. 21

[86] E. Hovy and C.-Y. Lin, "Automated text summarization and the Summarist system," in *TIPSTER TEXT PROGRAM PHASE III: Proceedings of a Workshop held at Baltimore, Maryland, October 13-15, 1998*, (Baltimore, Maryland, USA), pp. 197–214, Association for Computational Linguistics, Oct. 1998. 26

[87] R. Bommasani and C. Cardie, "Intrinsic evaluation of summarization datasets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8075–8096, 2020. 30, 31

[88] P. E. Shrout and J. L. Fleiss, "Intraclass correlations: uses in assessing rater reliability.," *Psychological bulletin*, vol. 86, no. 2, p. 420, 1979. 32

[89] T. K. Koo and M. Y. Li, "A guideline of selecting and reporting intraclass correlation coefficients for reliability research," *Journal of chiropractic medicine*, vol. 15, no. 2, pp. 155–163, 2016. 32

[90] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014. 40

[91] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, 2019. 40, 41

[92] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mt5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020. 40, 41

[93] M. Marreddy, S. R. Oota, L. S. Vakada, V. C. Chinni, and R. Mamidi, "Clickbait detection in telugu: Overcoming nlp challenges in resource-poor languages using benchmarked techniques," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021. 41

[94] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019. 41

[95] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013. 42