Islamic University of Technology (IUT)

ChartSumm: A large scale benchmark for Chart to Text Summarization

Authors

Raian Rahman, 170041014

Rizvi Hasan, 170041038

Abdullah Al Farhad, 170041042

Supervisor

Md. Hamjajul Ashmafee, Lecturer,

Dept. of Computer Science and Engineering

A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of B.Sc.

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
A Subsidiary organ of the Organization of Islamic Cooperation (OIC)
Academic Year: 2020-2021

May 17, 2022

Islamic University of Technology (IUT)

ChartSumm: A large scale benchmark for Chart to Text Summarization

Authors

Raian Rahman, 170041014

Rizvi Hasan, 170041038

Abdullah Al Farhad, 170041042

Supervisor

Md. Hamjajul Ashmafee, Lecturer,

Dept. of Computer Science and Engineering

A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of B.Sc.

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
A Subsidiary organ of the Organization of Islamic Cooperation (OIC)
Academic Year: 2020-2021

May 17, 2022

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Raian Rahman, Rizvi Hasan and Abdullah Al Farhad under the supervision of Md. Hamjajul Ashmafee, Lecturer of Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Gazipur, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Authors:

Raian

Raian Rahman
Student ID: 170041014

Rizvi

Rizvi Hasan
Student ID: 170041038

Farhad

Abdullah Al Farhad
Student ID: 170041042

Supervisor:

[illegible]

Md. Hamjajul Ashmafee
Lecturer,
Department of Computer Science and Engineering,
Islamic University of Technology

Dedicated to our parents and family members

Table of Contents

List of Figures

List of Tables

Acknowledgment

We would like express our gratitude towards IUT authority for providing assistance required to implement our proposed work. We would also like to give our heartiest thank to Tahmid Rahman Laskar from Dialpad Inc, Canada for his extensive support and guidance in our research. We are indebted to our professor, Dr. Abu Raihan Mostofa Kamal for providing us with insightful knowledge and guiding us at every stage of our journey. Finally, we would like to express our heartiest appreciation towards our family members for their continuous support, motivation, suggestions and help, without which we could not have achieved the scale of implementation that we have achieved.

Abstract

Information visualization such as bar- and line-charts are quite popular for understanding large tabular data. But, interpreting information solely with different visualization techniques can also be difficult due to different reasons like visual impairment or the requirement of prior domain knowledge to understand the chart. Automatic "chart to text summarization" can be promising and effective tool for providing accessibility as well as precised insights of chart data in natural language. In spite of having a good potential, there have not been a lot of works on chart to text summarization making it a low resource task. Scarcity of large scale datasets for chart to text summarization is one of the reason behind this. The human written descriptions in the available dataset also contains information beyond the knowledge of the chart making it difficult for us to have an unbiased evaluation. In our thesis, we propose *ChartSumm* a large scale dataset for chart to text summarization consisting of 84,363 charts along with their metadata and descriptions. We also propose two test sets: *test-e* and *test-h* for evaluating the performance of the trained models available in this domain. Our experiment shows that a T5 model trained on our dataset has achieved BLEU score of 75.72 in *test-e* set and 64.78 in *test-h* set. From our analysis we can conclude that large language models like T5 and BART can generate short precised deception from given chart metadata.

keywords — Chart summarization, Natural language generation, Low resource task

Chapter 1

Introduction

It is often said that "*We are drowning in Data yet Starving for Knowledge*". This is mainly because, even after generating a huge amount of data every moment, extracting knowledge from data is still a challenging task. The area of data mining or knowledge discovery mainly focuses on the nontrivial extraction of implicit and previously unknown useful information from data. In simpler words, data mining finds valuable information hidden in data.

1.1 The context

This is the era of big data. A huge amount of data is being generated every moment. Most of the data is being generated online. Most of the data is stored in tabular format. Although, The tabular format is the most structured way of storing data understanding the salient information from tabular data is a challenging task. This is mainly because there is no notion of trend or important highlights in the tabular data, it is quite difficult to find important information from tabular data [11, 12].

To solve this problem, different tools and techniques are used to easily interpret the hidden information inside data. Visualization techniques are the most common tool for finding hidden information from data [13]. Bar charts, pie charts, line charts, and scatterplots are some of the most commonly used visualization tools.

1.2 The problem

Although visualization tools like bar chart, line chart, pie chart can be useful tool for the users, it has its challenges as well. This is because people who are visually impaired cannot understand the visualization [7, 14]. Another reason is, we often need to visually compare between several graphical marks of a chart to infer key insights from data, which may be challenging. A previous study showed that text associated with the chart failed to convey any insight from that chart in 35% of the instances, while another 26% of the cases the text conveyed only a portion of the chart's intended message [15].

Natural Language Generation (NLG) for chart could be a solution to the problems stated above. In the chart to text summarization, important data and trends are depicted in a natural language summary. The generated text can be used in the text to speech system for visually impaired people who will be able to understand the meaning of the chart. Also, it can help the user in understanding and interpreting charts by conveying the key points about the chart by focusing on temporal, causal, and evaluation aspects [13]. It will also be easier to make comparisons in a natural language summary. Generated descriptions can also be used for indexing documents that contains charts for improving the performance of information retrieval algorithm [16].

In spite of having huge potential, automatic chart summarization systems are still in their infancy. Early works mostly focused on statistical methods for finding important information from chart and then planning based approaches for structuring captions [17–19]. Commercial systems like Quill [20] and Wordsmith [21] also uses the similar approach. Data driven approaches also gained popularity among researchers in recent times [1, 22]. For chart to text generation task two different data driven approaches were seen. One relies solely on the chart image making the task work like an image captioning task [9]. The other generates caption based on the chart metadata for example: chart table, title, labels etc [8, 10, 11]. Performance of models proposed on these systems show that models perform much better when trained with chart metadata

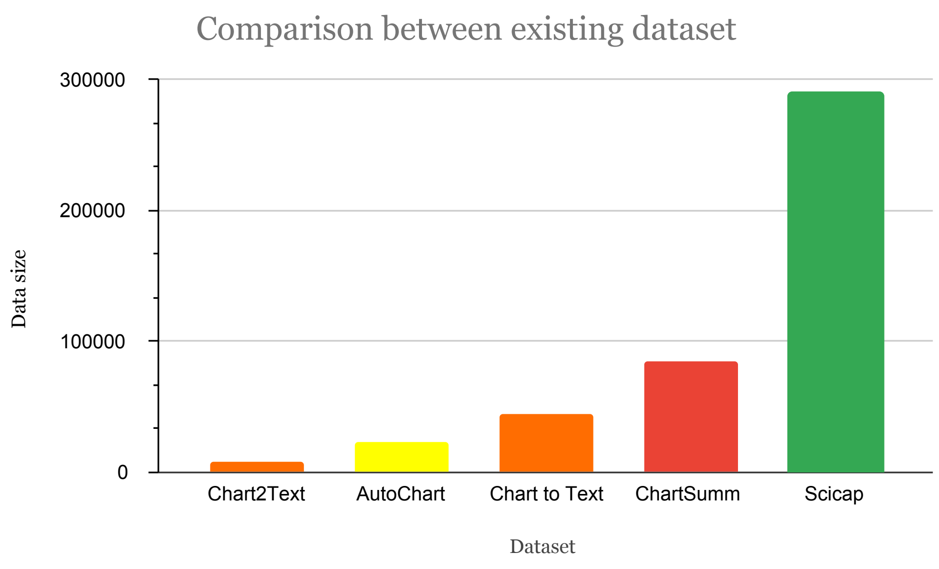


Figure 1.1: Comparison between existing dataset

compared to chart image.

Studies shows that there are two main challenges in chart summarization task. First being the scarcity of large scale dataset which makes it difficult to have a good performance using data driven approaches. To the best of our knowledge, we only have 4 datasets [8–11] proposed for chart to text summarization task. Only three [10, 11, 23] of them contain both chart image, metadata and a well defined summary. The largest of these dataset only contain around $44,085$ samples. Figure 1.1 shows the comparison between existing dataset. Also, the summaries proposed in both Chart2Text [11] and Chart-to-Text [10] are more descriptive and often contains information beyond the knowledge of

Second, there is also scarcity of strong baselines that utilizes the latest methods in natural language generation task. Although, recent works [1, 11] focused on achieving decent result by adapting transformer [24] based architecture. But none of them adapted any of the recent state of the art large scale language models.

1.3 Proposed Solution

In our thesis we wanted to address the two challenges stated in the previous section. We propose a large scale dataset for chart to text summarization comprising of $84,083$ chart images with corresponding chart metadata and well defined summary. We also propose two different test sets: *test-e* where the reference summaries contain information that can be derived from only the chart data while *test-h* which may contain longer chart description and some information in the chart might not be derived from the chart metadata. After collecting and processing dataset, we trained three available models and two large language models with our dataset and evaluated the performance. After evaluation we propose a strong baseline based on T5 architecture. To summarize, our contributions are:

1. Propose a new large scale dataset for chart to text summarization with human written summaries.

2. Two different test set (test-e and test-h) for evaluating a model's performance.

3. Based on extensive analysis, introduce a strong baseline for chart to text summarization task.

Chapter 2

Background Study

A survey of the existing literature in the field of our research, leading up to our topic, is included in this background study.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that deals with a program's understanding of human or natural language. It is a field that studies the relationship between data science and human language. Today, NLP is thriving as a result of massive gains in data access and computational capacity.

NLP can assist us with a wide range of tasks, and the fields of application seem to be expanding on a regular basis. For example: text classification, text segmentation, Named-Entity recognition, sentiment analysis, detection of disease, question aswering, machine translation, text summarization etc.

When we apply NLP on a text,initially it converts the input raw data into tokens which we call tokenization, and then the tokens go through a series of processes such as stop word removal, lemmatization etc. After that, we extarct features from the processed data using word embedding techniques. Then it supervises our input data using trained pipelines. Finally, after evaluation we deploy the corresponding task of NLP. Figure 2.1 represents the general pipeline of NLP.

2.1.1 Word Representation

In very simplistic terms, word embedding are the texts converted into numbers and there may be different numerical representations of the same text. As it turns out, many Machine Learning algorithms and almost all Deep Learning Architectures are incapable of processing strings or plain text in their raw form. They require numbers as inputs to perform any sort of job. So, a word embedding format generally tries to map a word using a dictionary to a vector. There are different types of word embedding:

- Count Vector
- TF-IDF vector
- Co-Occurrence Vector
- Continuous Bag of Words
- Skip-gram model
- Word2Vec

Among these techniques count vector, tf-idf and co-occurrence vectors are frequency based word embedding. They don't properly represent the meaning of each word. More specifically, we cannot get a good vectorized representation on different dimensions of each word represented using these techniques. But, C-Bow and Skip-gram [25] models are both developed by prediction based approach and these embedding contain the meaning of each word. They are briefly explained below:

Count Vector

This model counts the number of times a word appears in a document by first learning its vocabulary. Given D documents and N different words in the vocabulary, the size of the count vector matrix will be $D \times T$. The count vector matrix keeps track of the number of times a word occurs in documents. Given a large corpus, complexities may arise, stop words can be used(i.e removing common words like a, an, that) , or some top words can be extracted from the vocabulary based on frequency then use a new vocabulary or both methods can be used. Also in a large corpus, count data will

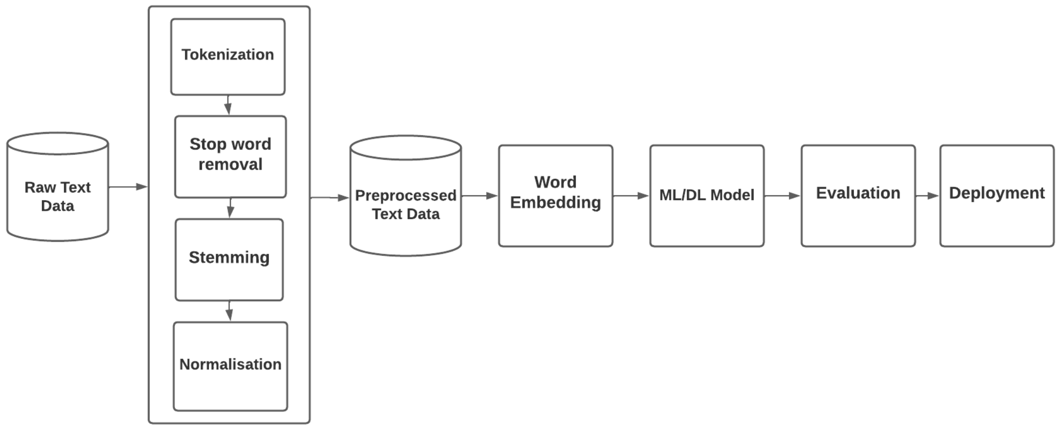


Figure 2.1: General pipeline of NLP

be composed of the more frequent words like; the, a, an , that overshadow the less frequent words which might provide meaningful information.

TF-IDF Vectorization

In order to re-weight the count features into floating point values suitable for usage by a classifier the tf–idf transform is used thus taking into account not just the occurrence of a word in a single document but in the entire corpus. Given an article on football, the article contains more sports-related terms like goals , in comparison to any other article.But words like “a , an, the” will appear frequently ; this method penalizes these types of high frequency words. Tf means term-frequency while tf–idf means term-frequency times inverse document-frequency.

Co Occurance Matrix with a fixed context window

It is inspired by the idea of **“A word is recognized by the company it keeps"** principle. Similar words tend to occur together and will have similar context. It preserves the semantic relationship between words

Continuous Bag of Words (CBOW) [25]

The way CBOW works is that it tends to predict the probability of a word given a context. A context may be a single word or a group of words. But for simplicity, I will take a single context word and try to predict a single target word.

Skip Gram Model [26]

Skip-gram follows the same topology as C-BOW. It just flips C-BOW’s architecture on its head. The aim of skip-gram is to predict the context given a word. Skip-gram approach was first introduced in 2013 by a paper by Mikolov. et. al [25]. An advanced version of this paper was introduced in 2014 by the same author. This paper used neural network to predict the next word in the text. The embedding found in skip-gram can easily capture the meaning of each word in different dimensions of the embedding.

GloVe word vector [26]

Word2vec is a combination of models used to represent distributed representations of words in a corpus C. Word2Vec (W2V) is an algorithm that accepts text corpus as an input and outputs a vector representation for each word. There are two flavors of this algorithm namely: CBOW and Skip-Gram. It is based on the paper [25]. Word2Vec can represent similar words in a similar dimension. Different versions of word2vec

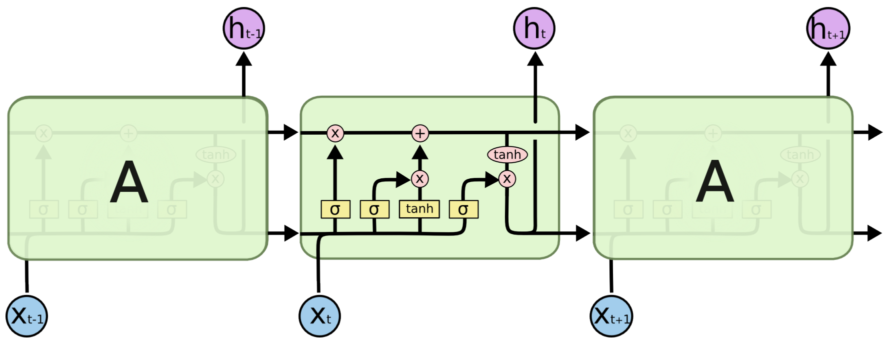


Figure 2.2: Repeating module in a LSTM network.

have been implemented. GlOVE [26] is a popular word embedding. It was build on 300 billion words and each word has 300 dimensional embedding.

2.1.2 Models

Most tasks in Natural Language Processing (NLP) has sequence as its input. Some of the tasks also focuses on generating a new sequence. Due to this reasons, state of the art models for different tasks of NLP has different architecture compared to other domains on deep learning. Some of the state of the art models on different tasks of NLP are explained in the following subsections.

LSTM [27]

Long Short Term Memory (LSTM) network is a type of Recurrent neural network (RNN) that can learn long term dependencies. Hochreiter and Schmidhuber (1997) [27, 28] proposed them first. LSTM network is specifically developed to prevent the problem of long-term dependency. They don't have to try hard to remember knowledge for lengthy periods of time. Recurrent neural network takes the form of a chain of repeated neural network modules. This repeating module in ordinary RNN will have a relatively simple structure, such as a single tanh layer.Now, LSTM has a chain-like structure as well, but the repeating module is different here. There are four neural network layers instead of one, each interacting in a unique way. Key part of LSTM is cell state. The cell state resembles that of a conveyor belt which helps to flow the information straight without any change. The ability of LSTM to remove or add information to the cell state is carefully controlled by structures known as gates. Gate allows us to choose whether or not to let information through. Every LSTM network has three gates to control the cell state. Following figure 2.2 represent the repeating module in a LSTM network.

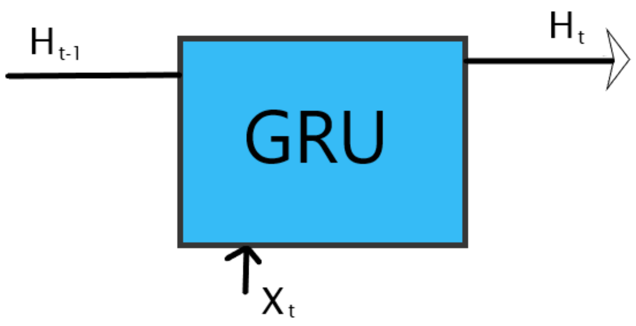


Figure 2.3: GRU architecture

GRU [29]

The Gated Recurrent Unit (GRU) is a step forward from the regular RNN. Long Short Term Memory and GRUs are quite similar (LSTM). GRU, like LSTM, controls the flow of information through gates. In comparison to LSTM, they are quite new. This is why they outperform LSTM and have a more straightforward architecture. Again, another intriguing feature of GRU is that, unlike LSTM, it lacks a distinct cell state. It only has one state: hidden. GRUs are easier to train because of their simpler architecture.

Let's look at how GRU works. Basically, GRU cell resembles an LSTM or RNN cell in appearance.It takes an input and the hidden state from the previous timestamp at each timestamp. It then produces a new hidden state, which is then passed on to the following timestamp.In contrast to an LSTM cell, which has three gates, a GRU now has only two gates(reset gate and update gate). The following figure 2.3 represents a GRU architecture.

Transformer [24]

In sequence data processing, we need to use recurrent neural network where the output of previous timestamp is also used as the input of the network. Most of the tasks in Natural Language Processing is seq2seq. Where the input is a sequence of words or frames and the output is also a sequence of words or frames. Advanced RNNs with attention techniques like LSTM [28] and GRU [29] are used to work with these sort of problems. But, a problem with these architectures are they cannot capture attention for long sequences. Also, they often suffer when generating new sentences. Also, these recurrent units work in sequential manner. So, parallelization couldn't be achieved causing the training process quite slow. A new architecture was proposed in Attention is All You Need paper [24]. The paper proposed Transformer architecture which solved

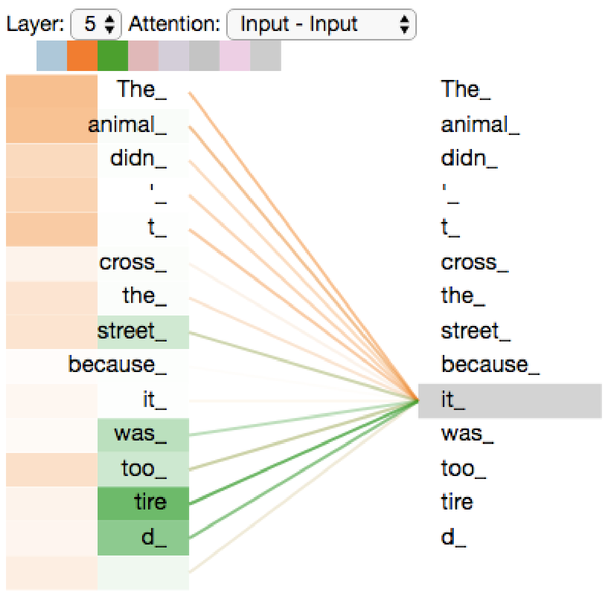


Figure 2.4: Attention of different words with “it" for the given example

all the problems with GRU and LSTM based architecture. Given enough resources, transformer architecture can give attention over a long sequence of words predicting which words are relevant to which word. Transformer proposed 3 new things. These are explained below:

- Self Attention
- Multi headed approach
- Positional Embedding

All of these are briefly explained below:

- **Self Attention :**
  Self-attention is slightly different comparing to what the name suggests. In simple words, self attention provides attention to all the word. Here, attention means, the relevancy of each word in the sentence given a word of the sentence. Self attention can compute the relevancy of each words with a word in the sentence. It is explained with an example. Let we have a sentence: "The animal didn’t cross the street because it was too tired". Now, the question is what part of the sentence "it" referred to in the sentence. Did it refer to The animal or the street. Self attention can provide us with a score of each word in the sentence that is relevant with it. Figure 2.4 illustrates after training, which word(s) were given the most attention. In the figure, we can see that The animal was given the most attention.

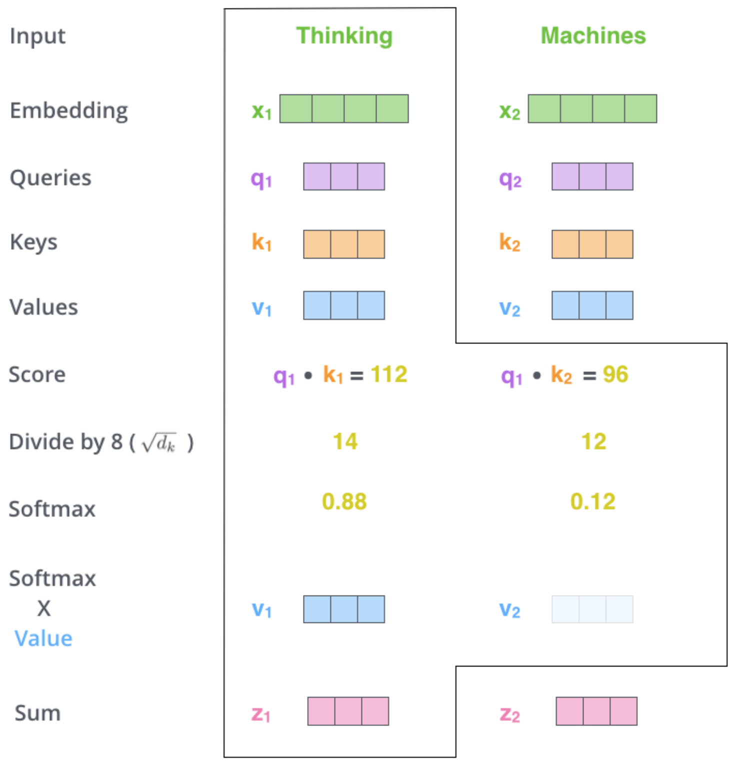


Figure 2.5: Process of attention calculation

Now, how attention is generated is also shown in figure 2.5. Here, we can see that in attention we have 3 different vectors. Query, Key and Value. If we multiply the embedding of the word with three weight vectors W^Q, W^A, W^V we will get Query, key and value vectors. Now, to calculate the score of each word, we need to dot multiply Query and Key vector.

$$Score_t = Q_t * k_t \tag{2.1}$$

Now, soft-max over all the words are calculated. Then the soft max value is divided by the square-root of the dimension of key vector to prevent gradient explode. Then the soft max value is multiplied with the value of each word is calculated and then summed. The sum is the output of self-attention layer at that key point.

$$Score = softmax(\frac{Q * K^T}{\sqrt{d_k}}).V \tag{2.2}$$

where Q,K and V are the query, key and value vector respectively. Then the output of self attention layer is sent to feed forward neural network.

- **Multi Headed Attention:**
  In transformer, the authors did not use a single self-attention module for each word. Rather, the authors show that if we have multiple self attention and concatenate the output of each self-attention layer, we can have a better representation. For the previous example, if we have multiple attention, the output for "it" word for the other words of the sentence is given in figure 2.6. In the paper, the authors used 8 multi headed attention on each layer.

- **Positional Embedding:**
  The core transformer does not have any recurrent unit. But, we already know that sequence data needs to have a recurrent unit. The authors of this paper proposed a brilliant method for this. They used positional encoding system to encode the position of each word in the sentence and that positional encoding provides us with a better representation.

- **Core Transformer Architecture:**
  The transformer uses encoder-decoder architecture for sequence to sequence modelling. The encoder and decoder architecture is illustrated in figure 2.7. Now, the input sequence (word embedding) is passed to the first encoder layer.

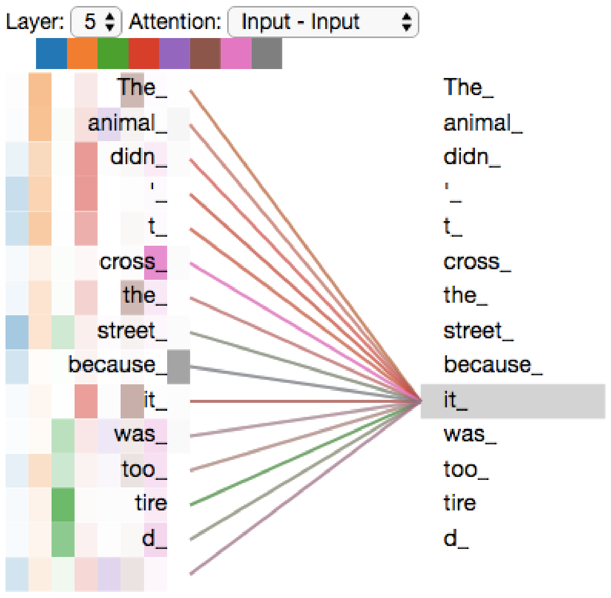


Figure 2.6: Output of multi headed attention layer. It gives a better representation as it gives attention not only to “the animal" but also to “was too tired"

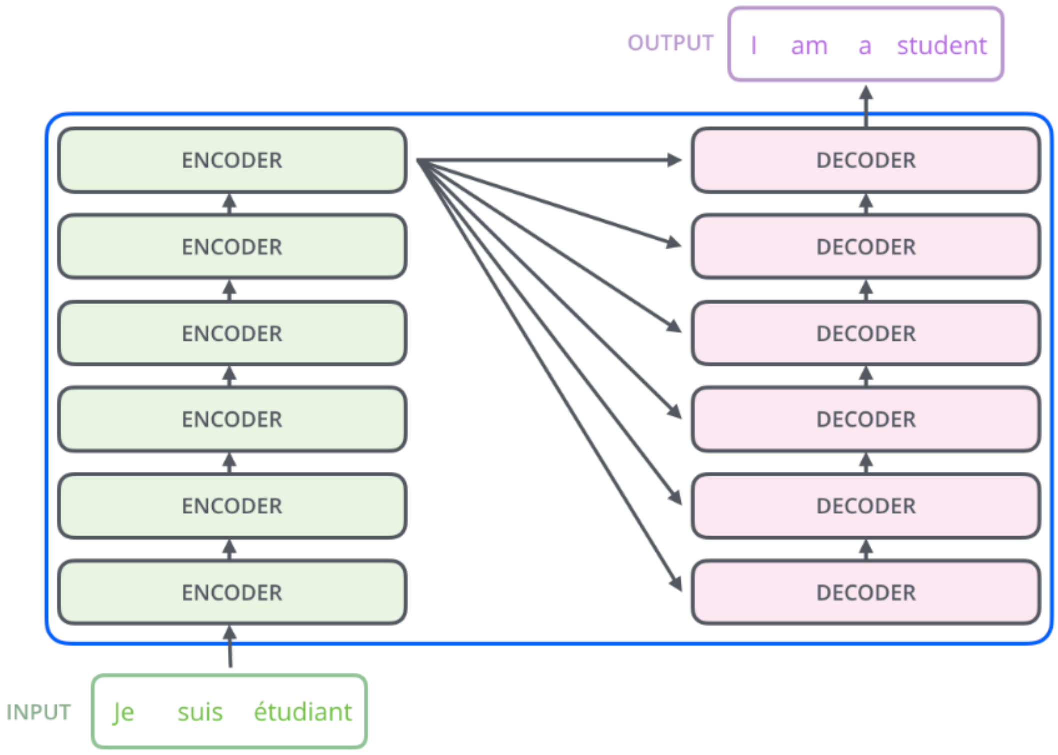


Figure 2.7: Transformer Architecture: 6 encoder block is stacked in encoder layer and 6 decoder block is stacked in decoder layer

Each encoder layer has two sub-block. Self-attention and Feed Forward Neural Network. The output of 6th or topmost encoder layer is passed through each of the 6 decoder blocks. The decoder block is same is the encoder block.

But, it has an extra encoder decoder attention block to have the context from source sentence. An overall architecture of transformer model is given in figure 2.8.

BERT [30]

After the introduction of transformer in [24], Natural Language Processing (NLP) has got a new boost. But, in those days, models were mostly taskcentric. In 2018, [30] proposed BERT, which introduced common pretraining in NLP. The acronym of BERT is **B**idirectional **E**ncoder **R**epresentation of **T**ransformer. It used Masked Language Modeling and Next Sentence Prediction for pretraining. Masked language modeling helped bert learn context from both direction. Next Sentence Prediction helped BERT learning to grasp the context of the sentence. With BERT, the idea of designing a generalized model was introduced. After pretraining, BERT could be fine tuned to any downstream task. In figure 2.9, a general architecture of BERT is shown.

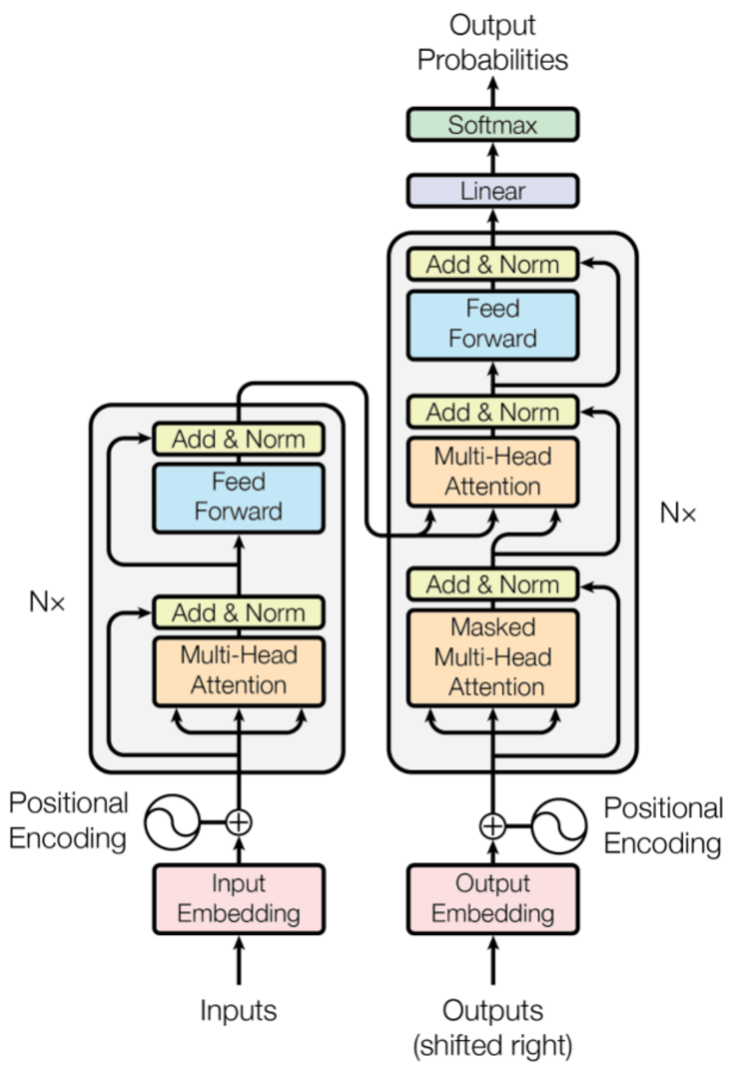


Figure 2.8: The figure illustrates transformer model. In an encoder block, the input embedding first passes through the multiple headed attention layer. Then it goes though feed forward neural network. In between each layer the output is added with previous output with skip connection and normalized. Then the output of encoder layer is passed to the encoder-decoder attention block of decoder. The decoder then decodes the context to a new sequence.

T5 [31]

Transfer learning, where a model is first pre-trained on a data-rich task before being fine tuned on a downstream task, has emerged as a powerful technique in natural language processing (NLP). T5 [31] introduced an unified framework that converts all text-based language problems into a text-to-text format. The pre-training was done on the “Colossal Clean Crawled Corpus” dataset that they introduced.

Text-To-Text framework

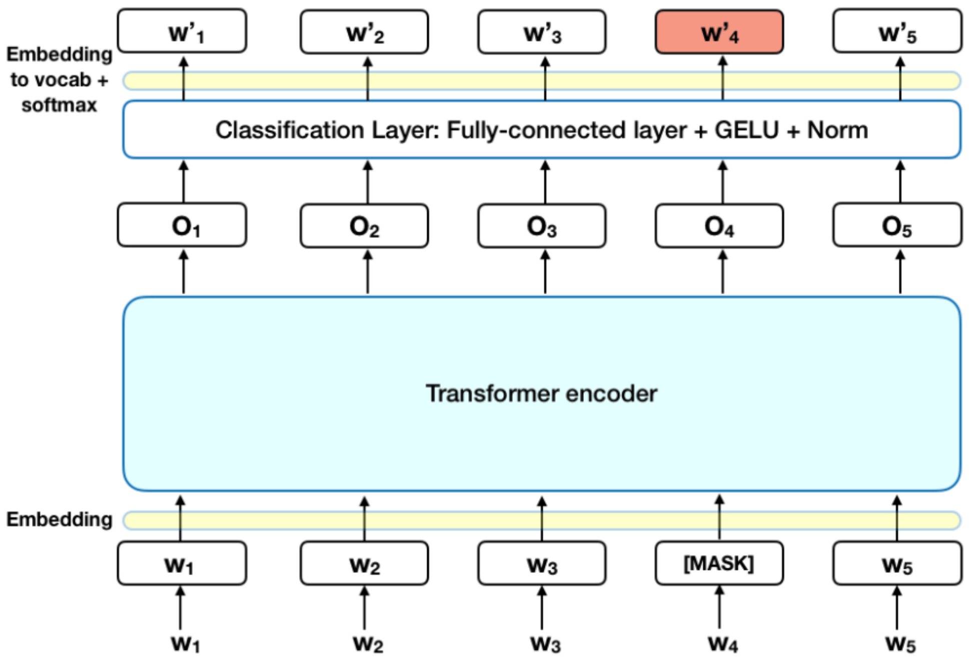
The input text should contain a prefix which states what the task should be, for example “translate to english”, followed by the text on which the operation should be performed. The output should contain the result of performing said operation on the text, and the output itself should be a sting. Even if the operation requires a float output, the float should be outputted as string. Refer to figure 2.10.

Extensions
T5 is flexible and can be extended to several tasks like language generation, machine translation, question answering, and many tasks that have not even been considered yet. Refer to figure 2.11.

BART [32]

BART [32] stands for bidirectional auto regressive transformer. It’s a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by (1) corrupting text with an arbitrary noise function, and (2) learning a model to reconstruct the original text. It is implemented as a sequence-to-sequence model with a bidirectional encoder over a corrupted text and a left-to-right autoregressive decoder. For pre-training negative log likelihood of the original document was used.

BART combines aspects from BERT which is an encoder only model, with aspects of GPT which is a decoder only model. In BERT random Tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for text generation. In GPT tokens



w'₁
w'₂
w'₃
w'₄
w'₅
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O₁
O₂
O₃
O₄
O₅
Transformer encoder
Embedding
w₁
w₂
w₃
[MASK]
w₅
w₁
w₂
w₃
w₄
w₅

Figure 2.9: General architecture of BERT

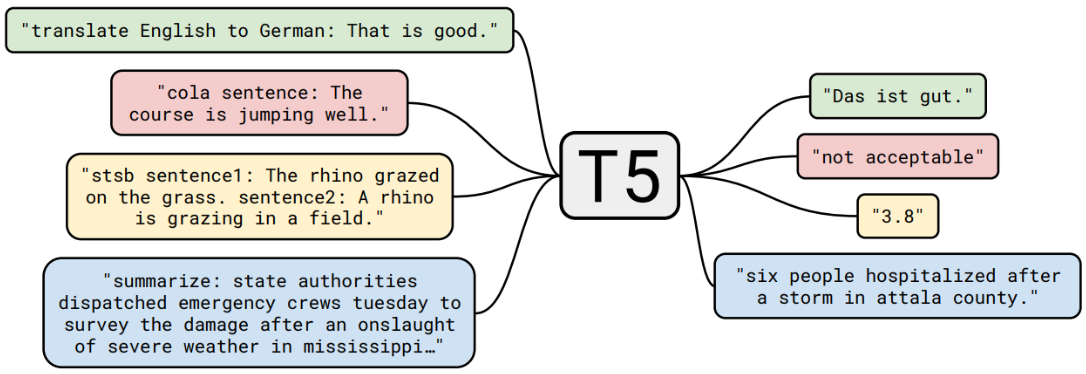

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 2.10: Diagram explaining T5's text-to-text framework.

are predicted auto-regressively, meaning GPT can be used for generation. However, words can only condition on leftward context, so it cannot learn bidirectional interactions. BART uses an encoder-decoder model where the encoder is similar to BERT and decoder is similar to GPT. Refer to figure 2.12.

Functions used to corrupt text

- **Token Masking:** Random tokens from the input are replaced with a [MASK] token.
- **Token Deletion:** Random tokens from the input are deleted
- **Text Infilling:** Random sequences of consecutive tokens are masked with a single [MASK] token.
- **Document Rotation:** A token is selected at random. The document is rotated such that the selected token is the first token. The task is to find the actual start

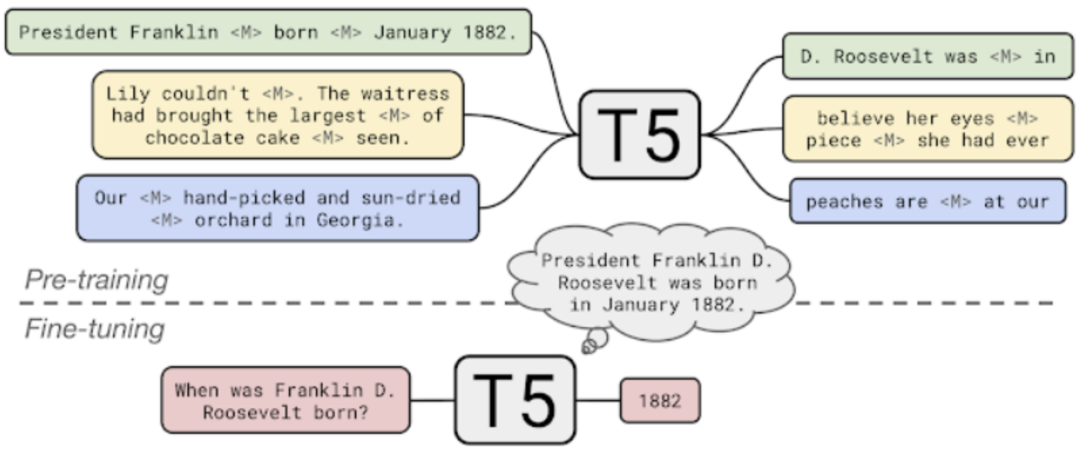

President Franklin <M> born <M> January 1882.
Lily couldn't <M>. The waitress had brought the largest <M> of chocolate cake <M> seen.
Our <M> hand-picked and sun-dried <M> orchard in Georgia.
T5
D. Roosevelt was <M> in
believe her eyes <M> piece <M> she had ever
peaches are <M> at our
President Franklin D. Roosevelt was born in January 1882.
Pre-training
Fine-tuning
When was Franklin D. Roosevelt born?
T5
1882

Figure 2.11: Example of how T5 can be expanded.

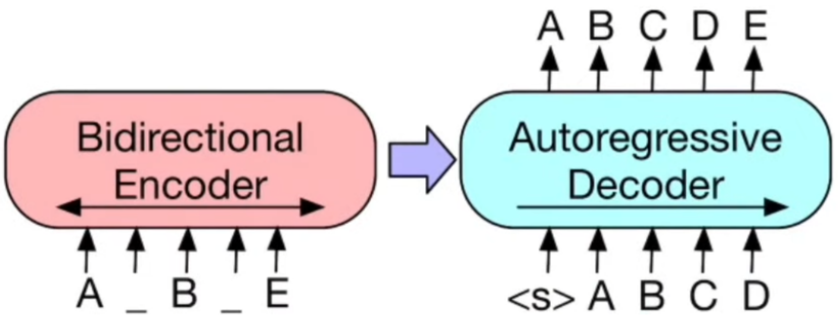


Figure 2.12: BART uses an encoder-decoder model where the encoder is similar to BERT and the decoder is similar to GPT.

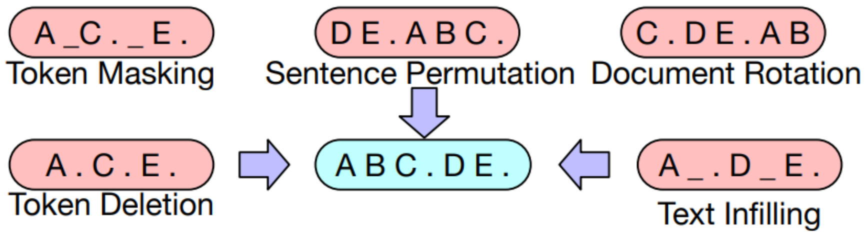


Figure 2.13: Types of text corruptions used in BART

of the document.

2.2 Data visualization

Graphical representations of information is referred to data visualization. By leveraging complex sets of numerical or factual numbers, visualization plays a significant role in data analytics and aids in the interpretation of large data in a real time.

Because of how the human brain processes information, using charts or graphs to display large amounts of complex data is more accessible than using structured infromation like spreadsheets and reports. Visualizations are a quick, natural, and simple approach to express essential concepts across multiple platforms. Moreover,These techniques allow us to experiment with different scenarios by making minor changes.

There are different types of visualization techniques. For example -

- **Bar Chart :** Bar chart uses rectangular bars with heights or lengths proportionate to the values they represent to convey categorical data. The bars can be plotted either horizontally or vertically.

- **Pie Chart :** A pie chart is a circular representation with slices to show numerical proportion. The arc length of each part in a pie chart is proportionate to the quantity it shows.

- **Map :** A map chart is a visual representation of items on a background that is frequently, but not necessarily, geographical.
- **Scatter Plot:** A scatter plot is a form of plot that displays values for two variables for a collection of numerical data using Cartesian coordinates.
- **Line Chart :** A line chart is a style of chart that represents data as straight line segments. It is a basic chart that is used in various fields.
- **Area Graph :** An area chart or graph depicts quantitative data graphically mainly based on line chart.
- **Bubble Chart:** A bubble chart is a form of graph that shows data in three dimensions.
- **Pareto Chart:** A Pareto chart is a graph that includes both bars and a line graph.

There are more visualization techniques to represent structured data. Those are used according to their simplicity and demand. Charts can be classified into two types:

- **Simple chart :** The chart which contains single variable in the y-axis is generally known as simple chart.
- **Complex chart :** A complex chart is one that has two or more variables on the y-axis.

Chapter 3

Literature Review

This literature review is an overview of the previously published works on our research topic. We have divided our literature review into 4 parts. (i) Data to Text Summarization systems, (ii) Chart data extraction systems, (iii) Chart to text summarization systems and (iv) Chart to text summarization datasets.

3.1 Data to Text Summarization Systems

In this kind of system, structured data are loaded as input to the system and finally as output we get summary of the corresponding data.

3.1.1 Enhanced Transformer Model for Data-to-Text Generation [1]

This paper was authored by Gong. et al. in 2019 [1]. It was published in the workshop on neural generation and translation. They used ROTOWIRE [33] dataset. In their work, they proposed a new transformer based architecture enhancing the base transformer model. In modification, they changed the input embedding and added a new learning objective for content selection. The model architecture is shown in figure 3.1.

By proposing this architecture, they achieved state of the art score on ROTOWIRE dataset.

Limitations:

The method itself had some lackings. It tends to generate terms that doesn't align with the context of the sentence.

3.1.2 Neural data-to-text generation with dynamic content planning [2]

This paper [2] was authored by K. Chen et al. and published in the Journal of Knowledge-Based Systems in Elsevier, April 2020. They used the ROTOWIRE [33] data-set. At the time of publication they outperformed the State of the Art over this dataset in

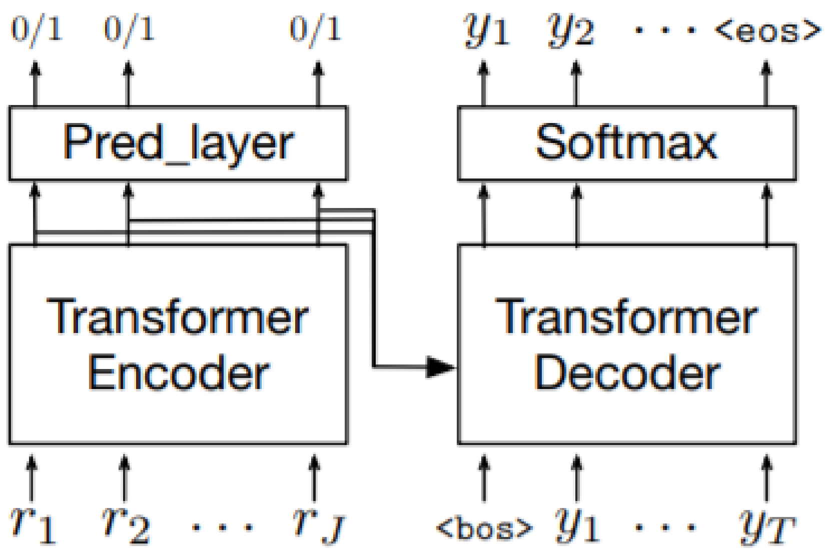


Figure 3.1: Model for "Enahnced Transformer" for data to text generation task

terms of of relation generation (RG), content selection (CS), content ordering (CO) and BLEU metric.

As shown in Figure 3.2, their model has 4 main sections:

- **Static Content Planning** acquires the selected records and their order which will be fed into the following component.For static content planning, they use the same mechanism proposed by [34].
- **Dynamic Content Planning** is the novel part proposed in this paper. It will decide which record will play an important role in generating the next word according to the current state.
- **Text Decoder** that generates word sequentially with attention [35] and copy mechanism [36] from the dynamic content plan representation.
- **Record Reconstruction** is the novel part proposed in this paper which encourages the decoder to generate more accurate information from the dynamic content plan representation. It will only be used in the training period.

Limitations:
The main limitation of this paper lies in the use of LSTM and RNN which are hard to parallelize and thus leads to very slow training.

3.2 Chart Data Extraction Systems

We searched for advanced systems to extract structured data from a chart. Here we found some works but the performance of those systems are not that much promising.

3.2.1 Tensor Field for Data Extraction from Chart Images: Bar Charts and Scatter Plots [3]

This paper [3] was authored by Jaya Sreevalsan-Nair, Komal Dadhich, Siri Chandana Daggubati and published in "Mathematics and Visualization, Cham: Springer Inter-

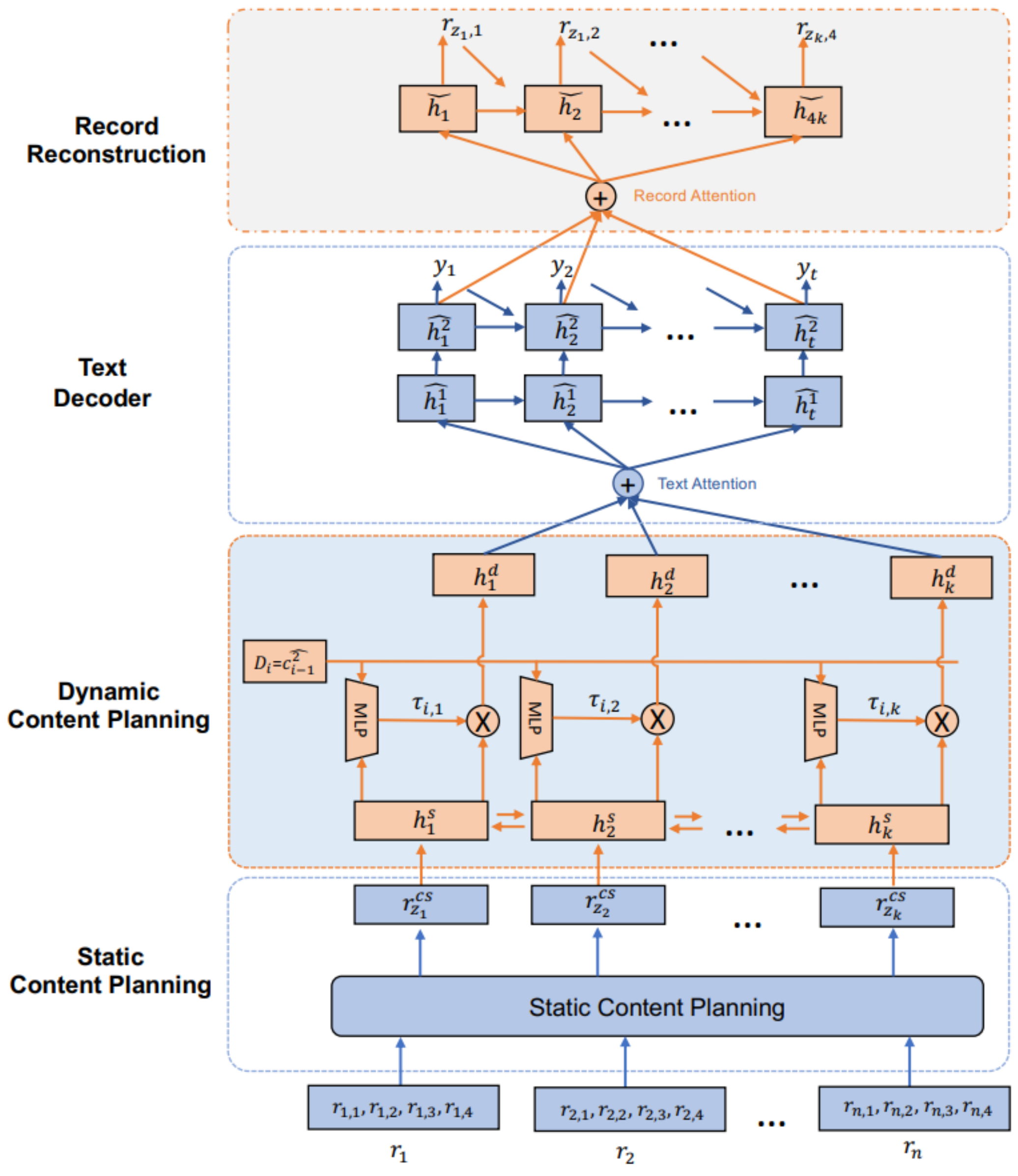
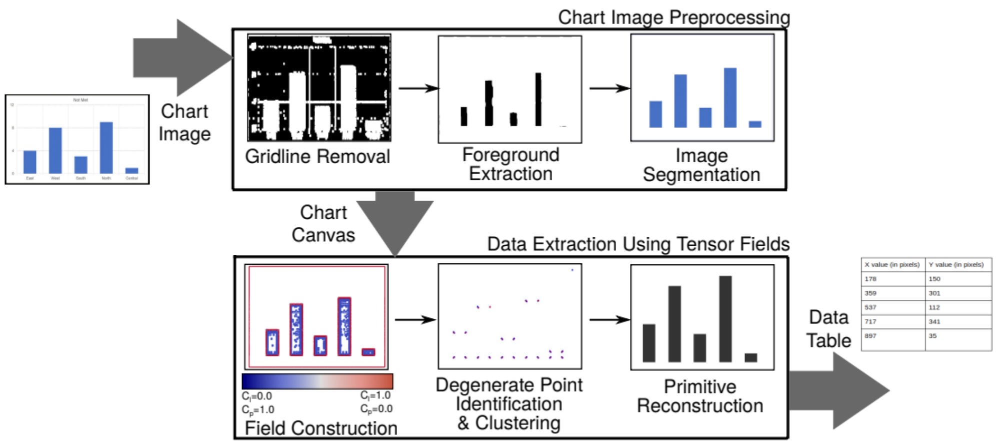


Figure 3.2: Model for "Neural data-to-text generation with dynamic content planning"



X value (in pixels)	Y value (in pixels)
178	150
359	301
537	112
717	341
897	35

Figure 3.3: Proposed workflow of data extraction from a chart image using positive semidefinite second-order tensor field of local geometric descriptors for "Tensor Field for Data Extraction from Chart Images: Bar Charts and Scatter Plots"

national Publishing", 2021. Their model ahieved Levels-A1 and A2 in Kimura's six-level scheme of statistical ability by performing data extraction with considerable accuracy.Their proposed solution was automatic and thus does not require user interaction during processing.

Figure 3.3 shows their proposed workflow. First they perform some image processing such as grid-line removal, foreground extraction, image segmentation. Then they perform data extraction using tensor fields, tensor field is constructed, degenerate points- i.e. 4 corner points of each bar for bar chart or each points for scatter chart, are identified and using degenerate point a primitive reconstruction is performed. Finally data table can be formed from the primitive reconstruction. They used traditional rule based algorithms for Chart canvas extraction and Data extraction using tensor fields from chart images.

Limitations:

There main limitations of this work is due to the tensor field being dependent on the user-defined image characteristics. These include image resolution and styling features of the plots, e.g., glyph shapes and sizes, bar width, and borders. Another limitation is that they only do it for bar chart and scatter plot.

3.2.2 ChartOCR: Data Extraction from Charts Images via a Deep Hybrid Framework [4]

This paper [4] was authored by J. Luo, Z. Li, J. Wang, and C.-Y. Lin and published in 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), 2021. They constructed their own dataset for this research. They conducted the experiment

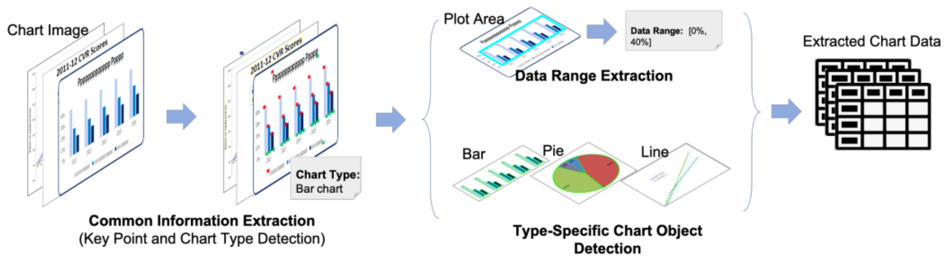


Figure 3.4: General workflow for "Chart OCR"

for Line chart, Bar chart and Pie chart. They combined traditional rule based method with Deep Neural Network to achieve an automatic system that does not require user input.

Figure 3.4 shows the general workflow of Chart OCR. First they extract type of chart and key points from the chart. Key point for bar chart are top left and bottom right points of each bar, for pie chart the key points are the center point, and points of each segment, for line chart the key points are points where gradient changes. Data is calculated using the type of chart and key points.

Figure 3.5 shows the structure of the model they used. The input image is passed through an hourglass network, then they are passed into two different convolution layer Conv 1 and Conv 2. Corner pool operation is performed on the output from the Conv 1 this is trained to predict key points. Max pool operation is performed on the output of Conv 2 and this is trained to predict type of chart. Finally, depending on the chart type a traditional rule based algorithm is used with the key points to calculate the data.

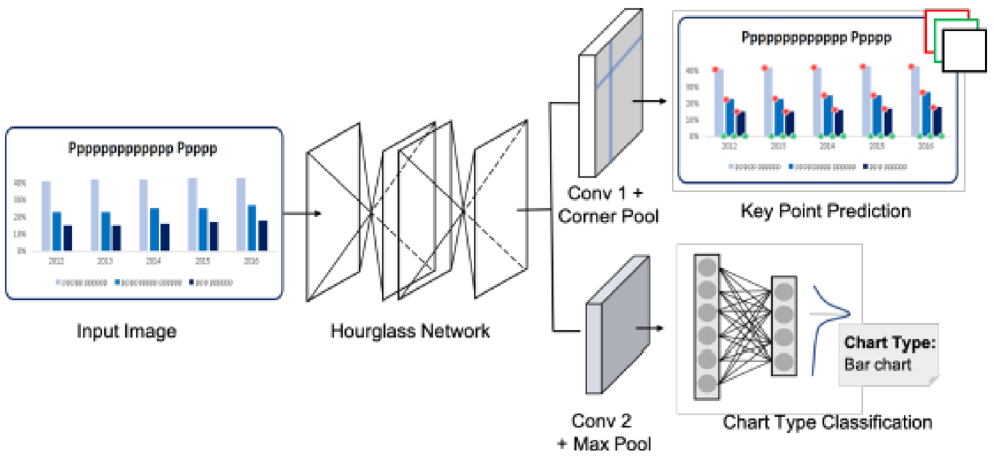


Figure 3.5: Abstract model for "Chart OCR"

Limitations:
The limitations of this model is that even charts of same type can vary dramatically. For example scale, position of text, style etc needs to be kept somewhat consistent.

3.2.3 Reverse-engineering Bart Charts using Neural Networks [5]

This paper [5] was authored by F.Zhou et al. and published in “Journal of Visualization, springer 2021”. They used neural networks to extract textual and numerical information from bar charts. They adopted a neural network-based object detection model. It can simultaneously localize and classify textual information. On the other hand, they designed an encoder-decoder framework to extract numerical information. They introduced an attention mechanism into the framework so that it achieves high accuracy and robustness. They evaluated their mechanism on both real-world and synthetic data.

Extraction of textual information:

They used Faster-RCNN to simultaneously localize and classify textual information which is a neural network-based object detection model. It reduces the use of traditional low-level operations for object detection to extract textual information. Figure 3.6 shows the general workflow of the extraction of textual information.

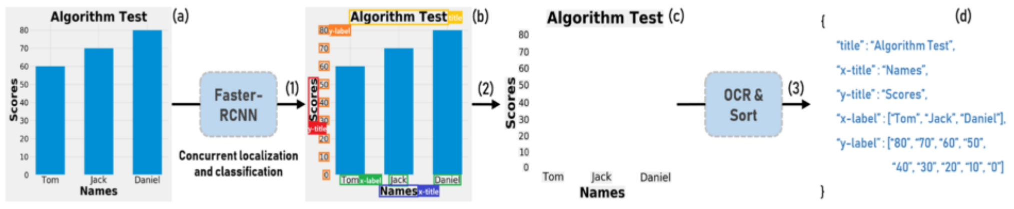


Figure 3.6: General workflow for extraction of Textual information

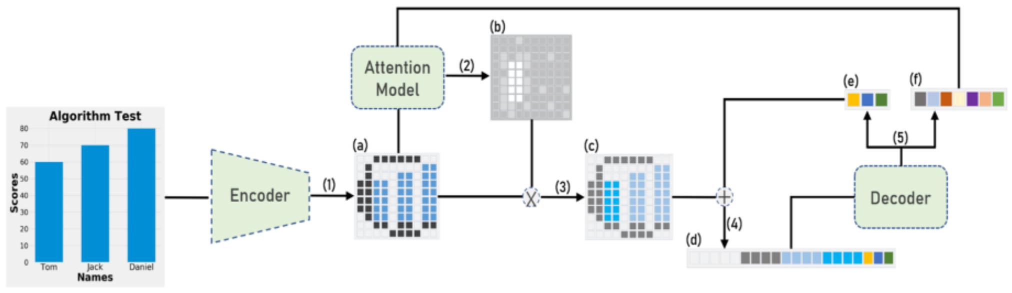


Figure 3.7: General workflow for extraction of Numerical information

Extraction of numerical information:

In figure 3.7 they used three neural network models to extract numerical information; the encoder, the decoder, and the attention mechanism. Basically, the encoder bar chart image as input and performs various operations to extract features. It then produces a sequence of feature vectors. After that, the attention model takes the feature vector and the hidden state of the decoder from the previous iteration and generates an attention vector. The attention vector and feature vectors are then combined and concatenated with the bar vector and the decoder iteratively generates the numerical information.

Limitations:

The training is not end-to-end since some works can be repeated. They have two separate parts that are trained separately which increases the running time. They only use bar charts.

3.2.4 BarChartAnalyzer: Digitizing Images of Bar Charts [6]

This paper [6] was authored by K. Dadhich et al. and published in the proceedings of the International Conference on Image Processing and Vision Engineering, 2021. They proposed a semi-automated workflow for data extraction from digitizing images of bar charts. They used second-order tensor fields from tensor voting in computer vision. They identified appropriate state-of-the-art algorithms for text recognition in bar charts. They created a dataset covering all seven subtypes. Figure 3.8 shows the seven main components of their workflow.

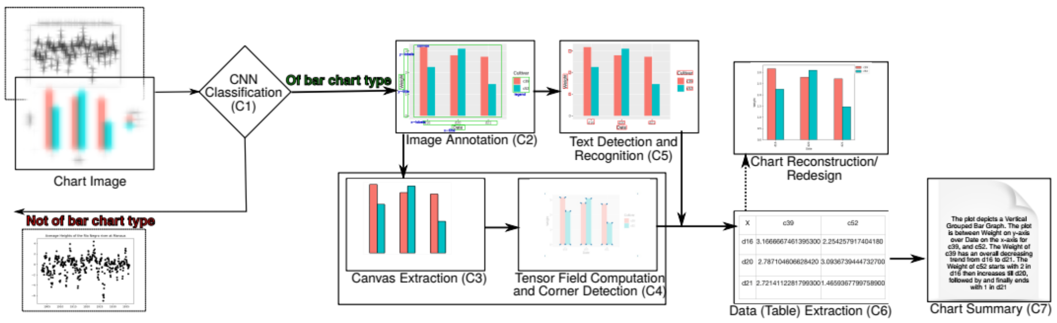


Chart Image
CNN Classification (C1)
Of bar chart type
Not of bar chart type
Image Annotation (C2)
Text Detection and Recognition (C5)
Chart Reconstruction/ Redesign
Canvas Extraction (C3)
Tensor Field Computation and Corner Detection (C4)
Data (Table) Extraction (C6)
Chart Summary (C7)

X	c39	c52
d16	3.166667461395300	2.254257917404180
d20	2.787104606628420	3.093673944473270
d21	2.721411228179930	1.465936779975890

The plot depicts a Vertical Grouped Bar Graph. The plot is between Weight on y-axis over Date on the x-axis for c39, and c52. The Weight of c39 has an overall decreasing trend from d16 to d21. The Weight of c52 starts with 2 in d16 then increases till d20, followed by and finally ends with 1 in d21

Figure 3.8: Semi-automated workflow for BartChartAnalyzer

Their workflow has seven main components.

Chart Sub-type Classification (C1):

Their classifier is inspired by the VGGNet (Visual Geometry Group Network) architecture. It is widely used for object detection and segmentation task on images.

Image Annotation (C2) and Canvas Extraction (C3):

To provide different labels to different regions of interest (ROI) in the Images, they use LabelImg as a tool to mark and annotate bounding boxes. The canvas extraction step includes image preprocessing methods to remove the remaining element other than chart objects.

Tensor Field Computation (C4):

They use local geometric descriptors as a second-order tensor for tensor vote computation.

Text Recognition (C5) and Information Aggregation for Data Extraction (C6):

They explore Character Region Awareness for Text Detection(CRAFT) for effective text area detection. They transform the data extracted in pixel space to data space and add appropriate textual information for the variable name and bar width to extract the data table.

Chart Summarization (C7):

They then generate the chart summary using the template from the previously generated data table.

Limitations:

This method suffers from errors in detection. It also generates false positives for corner detection in relatively low-fidelity images. Their classification model cannot handle variants of bar charts with textures in the bars or hollow bars. They only work on bar charts.

3.3 Chart to Text Summarization Systems

Chart to text summarization is still in its infancy. Although, the topic is getting interests from NLP researcher community. Structured data and chart images are placed into this type of system as input, and we get a summary of the associated data as output.

3.3.1 Chart-to-Text: Generating Natural Language Descriptions for Charts by Adapting the Transformer Model [7]

This paper [7] was authored by Jason Obeid, Enamul Hoque and was published in the Proceedings of the 13th International Conference on Natural Language Generation, November 2020.

They created their own dataset by scrapping data of bar charts and line charts from statista. Statista also contained summaries for those charts. The dataset contains data of 4466 bar charts and 3839 line charts. Let chart summery $CS = [I, D, T, L, S]$, where for each chart summery CS there is a chart image $i \in I$, data table $d \in D$ and

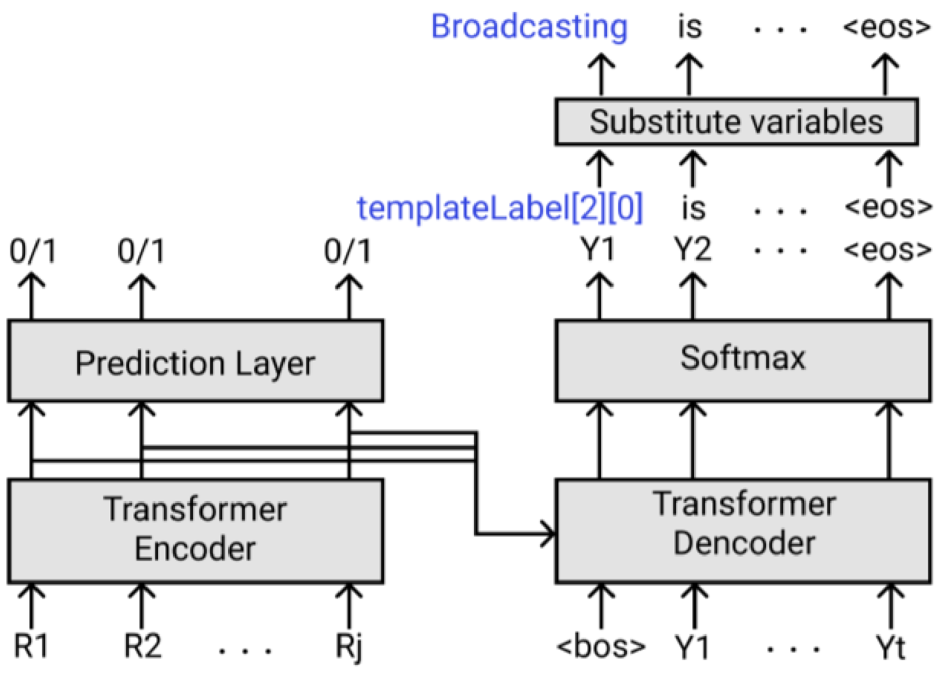


Figure 3.9: Abstract model for “Chart to text"

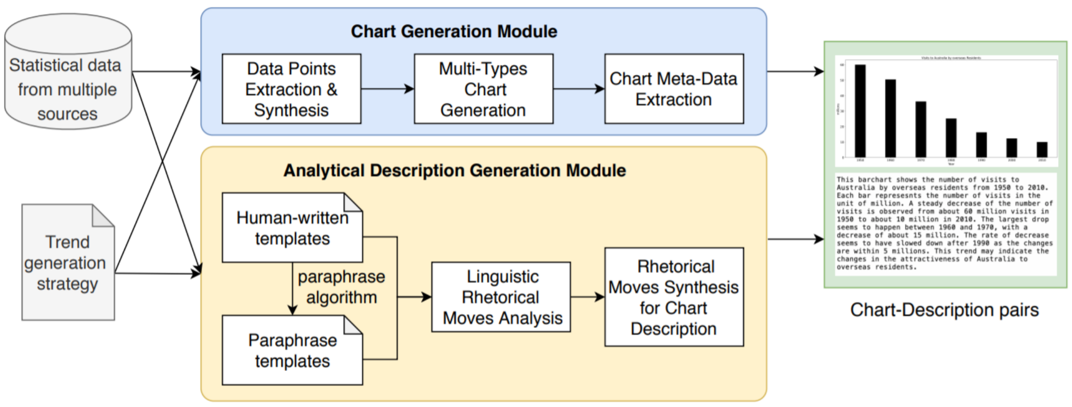
summery $s \in S$. Note that, while the dataset does contain chart images, they don’t use the chart images to train their model.

Their base model extends the standard transformer model [24] by adding a binary prediction layer and a content selection training step.The input layer of the model accepts a tuple of four features (entity, type, value, information) as input. The model then generates the latent representation of each record in the input sequence and passes it through the binary prediction layer. The decoder of this model is the same as the original Transformer model and predicts the summary based on encoder’s output. The model also removes the positional embedding of the transformer encoder, as there is no ordered relationship within the records of their dataset.

Input embedding: For each data table $d \in D$ they preprocess it into record $r \in R$. Each of these records are placed in a tuple with four features as follows:

- $r_i(0)$ contains the column header
- $r_i(1)$ contains the table cell value
- $r_i(2)$ contains the column index, and
- $r_i(3)$ contains the chart type

Figure 3.9 shows the **abstract model** of chart to text. The inputs $R1$ to RN are passed through the encoder. The prediction layer does content selection. Then the


Statistical data from multiple sources
Trend generation strategy
Chart Generation Module
Data Points Extraction & Synthesis
Multi-Types Chart Generation
Chart Meta-Data Extraction
Analytical Description Generation Module
Human-written templates
paraphrase algorithm
Paraphrase templates
Linguistic Rhetorical Moves Analysis
Rhetorical Moves Synthesis for Chart Description
Year
Chart-Description pairs

This barchart shows the number of visits to Australia by overseas residents from 1950 to 2010. Each bar represesnts the number of visits in the unit of million. A steady decrease of the number of visits is observed from about 60 million visits in 1950 to about 10 million in 2010. The largest drop seems to happen between 1960 and 1970, with a decrease of about 15 million. The rate of decrease seems to have slowed down after 1990 as the changes are within 5 millions. This trend may indicate the changes in the attractiveness of Australia to overseas residents.

Figure 3.10: Abstract model for “Autochart"

selected content is passed through decoder. The output of the decoder is passed through softmax to convert output to probabilities. Then the output is converted to tokens.

Limitations:
They only included line charts and bar charts in the dataset. The output gave us description rather than precise summaries of the dataset. Human written summaries in “AutoChart” dataset contains summary beyond the data of chart.

3.3.2 AutoChart: A Dataset for Chart to Text Generation task [8]

This paper [8] was authored by Jiawen Zhu1, Jinye Ran, Roy Ka-wei Lee1, Kenny Choo1, and Zhi Li2 and was published in Proceedings of the Conference Recent Advances in Natural Language Processing - Deep Learning for Natural Language Processing Methods and Applications, 2021.

They used their own dataset named Autochart for this experiment. They outperformed chart-to-text on this dataset. They propose a novel framework to generate charts and their corresponding analytical descriptions automatically. They also conducted extensive human and machine evaluation on the generated charts.

Figure 3.10 shows the abstract model for autochart. They scrapped data for their dataset from multiple statistical sources. They then extracted data points and formed charts. Then they extracted the chart metadata from the charts. They annotated those data manually using human linguists. Through a paraphrasing algorithm they generated paraphrase templates. Then through rhetorical move analysis they generated the summaries.

Limitations:
Generated summaries in the dataset are template based summaries, thus out of domain inputs do not perform well.

3.3.3 SCICAP: Generating Captions for Scientific Figures [9]

This paper [9] was proposed by Ting-Yao (Edward) Hsu, C. Lee Giles, Ting-Hao 'Kenneth' Huang and it was published in Findings of the Association for Computational Linguistics: EMNLP 2021.

In this paper, the authors proposed an end-to-end framework based on neural approach for generating informative high quality captions for scientific figures in academic literature. They also proposed a large scale dataset for caption generation. The dataset was collected by collecting arxiv papers. SCICAP contains around $2,90,000$ captions and summaries. The baseline proposed by SCICAP dataset uses a CNN based ResNet [37] encoder and LSTM [27] decoder for generating the caption. CNN based ResNet encoder first encodes the image into a sequence which is then fed to the decoder for generating the caption.

Limitations:

The key limitation of SCICAP dataset is it does not provide chart metadata along with the image and the caption. Also, our analysis of SCICAP dataset shows that most images from SCICAP dataset are vague and don't contain any meaning. In addition to that, the performance of the baseline model proposed by the paper is quite poor. It also indicates that just using image and summary for training can be a bottleneck for chart captioning or chart summarization system.

3.3.4 Chart-to-Text: A Large-Scale Benchmark for Chart Summarization [10]

This paper [10] was authored by Shankar et el and is accepted to be presented at the 60th Annual Meeting of the Association for Computational Linguistics (ACL), 2022. Only the preprint of this work was available at the time of writing this report and we found it just three weeks earlier to the time of writing this report. So, our work has not been directly influenced by this paper.

The paper proposes a large scale dataset for chart to text summarization. The dataset consists of chart image, corresponding metadata and human written description of the image. Their dataset contains around $42,000$ summaries along with the corresponding metadata and summary. In addition to this, they also proposed a strong baseline for their dataset. They experimented with BART, T5, Chart2Text models. Their analysis shows T5 provides the best result in chart summarization with highest BLEU, BLEURT score with lowest perplexity.

Limitations:

Just like chart2text dataset [7], Chart to Text dataset also collected their dataset from statista. As previously stated, statistics from statista often contains summary beyond the scope of information given in the chart. So, the raw performance of the model can not be checked using only chart to text dataset.

3.4 Chart to Text Summarization Dataset

Summarizing charts to text is still in development. Nonetheless, the topic is generating interest among NLP researchers. Despite of lacking chart to text summarization dataset, we examined each one.

3.4.1 Chart2Text [11]

Chart2Text is a dataset for chart to text summarization. The dataset consists of chart image, corresponding metadata and human written description of the image. Their dataset contains around 8305 summaries along with the corresponding metadata and summary. They created their own dataset by scrapping data of bar charts and line charts from statista. Statista also contained summaries for those charts. The dataset contains data of 4466 bar charts and 3839 line charts. Their dataset contains simple and complex charts.

3.4.2 Autochart [8]

There are 23,543 chart examples in Autochart. They gathered data samples from the World Bank dataset, the Food Nutrition dataset, and other sources. Each sample includes a chart image, metadata, and an explanation of the data. Their summaries are based on templates.

3.4.3 SCICAP [9]

Scicap, a chart captioning dataset, was proposed in 2021. It was collected from scientific journals. Despite the fact that chart captioning is comparable to chart summarization, chart captions in scicap do not have a lot of overlap between the chart image and the chart caption. Furthermore, generic captions make for more than 60

3.4.4 Chart-to-Text [10]

Chart-to-Text dataset provides chart metadata, chart image and gold summaries for each of the chart. This dataset provides real world data. Around 42, 000 summaries are included in their collection, together with the accompanying metadata and summary. They found two relevant sources with sufficient quantities and types of charts with textual descriptions after searching through numerous sources such as news sites, textbooks, and websites containing data facts. Those two sources are statista website

Table 3.1: Existing dataset for chart to text summarization

Dataset	Data source	Formulation	Summary Type	Example Count
RotoWire	NBA Game Summaries	Data table, Summary	Human written	$4,863$
Scicap	Scientific papers	Chart image, Chart summary	Captions of chart in scientific papers	$2,90,000$
chart2text	Statista	Chart image, Chart metadata, Chart summary	Human written	$8,305$
Autochart	World bank dataset, food nutrition dataset etc.	Chart image, Chart metadata, Chart summary	Template based summary	$23,543$

and pew websites. From statista they collected 34811 examples and from pew they collected 9285 examples. Their dataset conatins simple and conmplex charts. Their dataset consists of barchart, linechart, areachart, scatter chart, pie chart and table.

3.5 Comparison of existing dataset

Although, several benchmark datasets that are publicly available for data to text summarization task [33, 38, 39], there have not been a lot of publicly available dataset for chart to text summarization task. In table 3.1, publicly available datasets for chart data to text summarization is shown.

Although, there are couple of publicly available dataset for data to text summarization [33, 39], which are mainly based on sports domain and has large data table and its corresponding summary. These data can only be used for pretraining a content selection system for chart summarization but not for chart summarization directly. In 2021, Scicap [9] was proposed which is a chart captioning dataset. It was collected from scientific literature. Although, chart captioning is similar to chart summarization, captions of charts in scicap don't contain a lot relevancy between chart image and chart caption. Also, more than 60% of the captions of scicap contains generic captions.

To the best of our knowledge, there is only three publicly available chart to text summarization [7, 8, 10]. Chart2text and Chart to Text summarization dataset contains chart image, chart metadata and human written descriptions of chart collected from a statistical analysis website statista. Although, the dataset contains a decent amount of statistics, recent state of the art data hungry deep learning system's performance may be bottlenecked by the low amount of data in this dataset. Another available dataset autochart [8] used rhetorical question analysis and predefined templates and paraphrasing algorithm to a large scale chart to text summarization task. But, as the dataset is template based, it lacks from variation in chart description. Also, rhetorical question analysis cannot detect complex trends causing the quality of the description being poor.

Chapter 4

Proposed Methodology

As stated earlier, we propose a new large scale benchmark dataset called **ChartSumm**. We selected sources to collect data from, collected the data, cleaned and preprocessed the data, and ran benchmark using several models on the data to perform an exhaustive analysis.

4.1 ChartSumm Dataset

The key contribution of our thesis is “ChartSumm" dataset which is a large scale dataset for chart to text summarization task. In this dataset we compiled more than $80,000$ charts along with their metadata and well defined summaries which is double of the current largest dataset available for chart to text summarization. In figure 4.1, the pipeline of dataset construction is shown.

This section contains data collection, data processing, data splitting followed by dataset analysis.

4.1.1 Dataset Collection

To create a large dataset, first we need a data source. So, the first stage of our work was to find out appropriate source for our dataset. While there are a good amount of publicly available repository with statistics with charts are available, only very few of them contained summaries. It was possible to collect the statistics and generate gold summaries for those statistics. But available crowd sourced works for writing natural language descriptions are costly. So, we tried to select source where statistics and their short descriptions are available.

After exploring different sources including textbooks, academic literature, data repository and news websites, we have chosen two sources that seemed appropriate for our work. These sources are explained below:

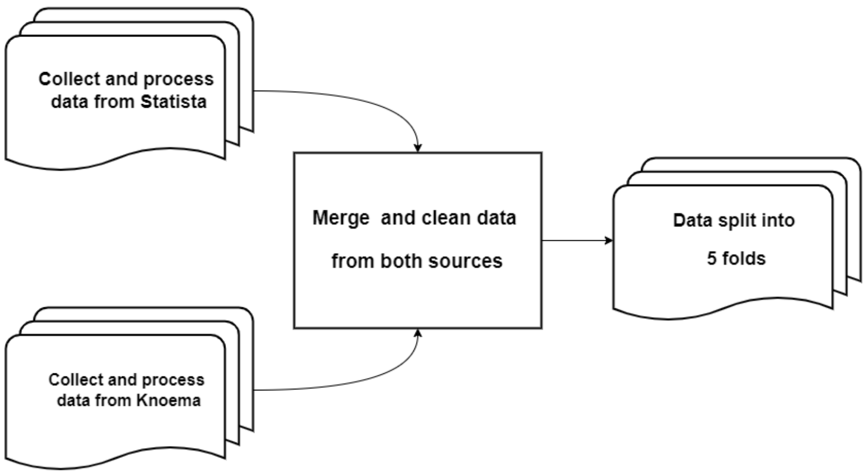

Collect and process data from Statista
Merge and clean data from both sources
Data split into 5 folds
Collect and process data from Knoema

Figure 4.1: Contruction pipeline of ChartSumm

Knoema

Knoema [40] is a statistical service based online platform. In their data atlas, they have a statistical collection of over $1,000$ economic indicators of more than 200 countries on earth. For each of the statistics, knoema generated a short description of the statistics. First, we crawled over $1,10,000$ statistics from knoema. After that, we filtered out the data sources that are not publicly available. It resulted $43,179$ publicly available statistics where the data is collected from that data source and description was taken from knoema website. All of the data available on knoema data atlas is simple chart. As the statistics are shown with respect to year, it can be considered as line chart. So, we classified each chart as simple line chart from this source.

Statista

Statista [41] is an online platform where statistics on wide range of topics are published along with a short human written description of the statistics. Topics in statista includes economics, marketing, industry and opinion research. First, we crawled over $7,50,000$ available pages in statista research to collect the list of publicly available statistics for our study. On March 2022, we collected around $50,000$ statistics, yielding a total of $41,184$ different charts along with summary and chart metadata. For each of the chart, first we took screenshot of chart image, but in statista around $4,500$ pages didn't include a chart image. So we generated the charts by ourselves using matplotlib. To decide on the type of charts, we used some heuristics to select the chart type while generating the charts. After collecting the data table, we classified the data into simple charts and complex chart depending on the number of columns in the chart.

4.1.2 Dataset Processing

Collecting this huge amount of data manually would have been time consuming. So we decided to build an automated system collecting data from knoema and statista. For both of the websites, we need to create a session before download any statistics. Both of the websites has anti automated use system enabled. So just writing a script to collect the dataset would have been resulted in banned account. As a result we needed to randomize the behavior of our data collector.

First, the data collector would create a new session. After that we selected a random number from 75 to 100. After downloading the specified number of data, the system would close the current session and go for a back off. The back off would be a random amount of time from 200 seconds to 400 seconds. After back off, the system would create a new session and start downloading the data again.

Although, the whole system was able to prevent getting capcha and detected by the websites, it made the whole process quite slow. On an average, we needed around 32 days to collect the data from these sources.

Statista

The list of steps taken to prepossess scraped data from statista are detailed below.

First we had to **Classify types of charts** To classify the charts into bar chart, line chart and pie chart we used ChartReader[1]. ChartReader has 84.99% accuracy for classifying the charts. ChartReader classifies the chart into Area Graph, Bar Graph, Box Plot, Bubble Chart, Flowchart, Line Graph, Map, Network Diagram, Pareto Chart, Pie Chart, Scatter Graph, Tree Diagram, Venn Diagram. Using different heuristics and analysis, we finally classified Area Graph, Line Graph and Map into Line Graph. We also combined Bar Graph, Box Plot, Bubble Chart, Network Diagram, Pareto Chart, Scatter Graph into Bar Graph. We kept Pie Chart as it is. Table 4.2 shows distribution of chart with respect to chart type for both knoema and statista.

Then we had to **Find missing x labels** Some of the scrapped data had missing x labels. So had to manually identify them using the following methods.

- **Year:** If all x values were integers less than 2050 and greater than 1800 we set the x label to “year".
- **Month:** If all x values were names of months, we set the x label to “month".
- **Day:** If all x values were names of days(saturday, sunday ...), we set the x label to “month".

[1]https://github.com/Cvrane/ChartReader

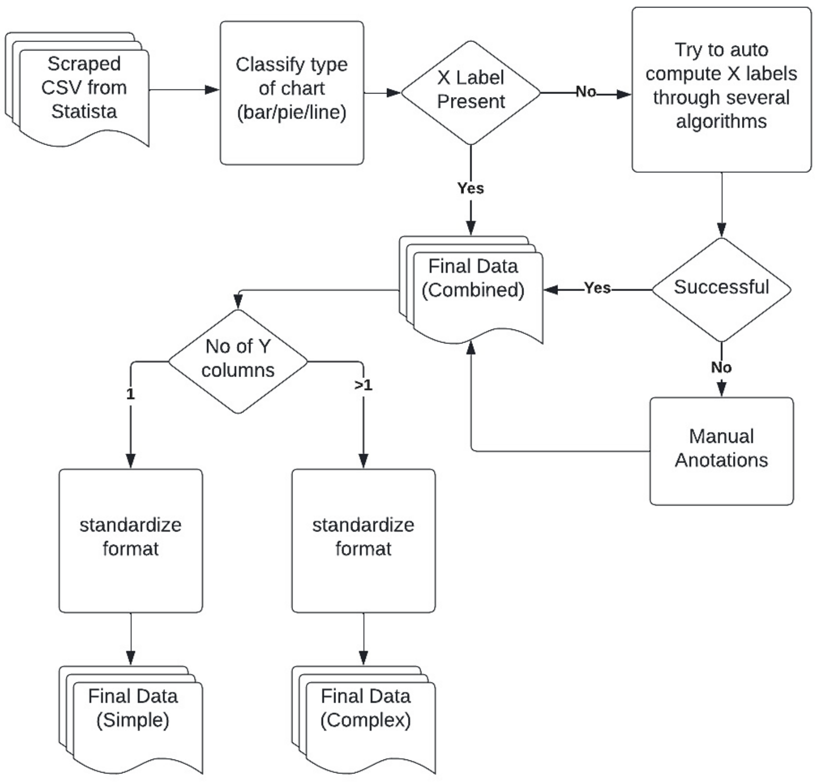


Figure 4.2: Figure showing the general steps involved in preprocessing scraped data from statista

- **Quarter:** If most x values were Q1/Q2/Q3/Q4 and optionally followed by an integer(year) we set x label as “quarter".
- **Country:** If more than 30% x values contained values from the list of all countries collected from Wikipedia, we labeled the x label as “country"
- **City:** If more than 30% x values contained values from the list of cities collected from World City Database, we labeled the x label as “city"
- **Area:** If the x labels contained the names of general areas like continent names, sub continent names, etc, we set the x label as “area".
- **NER:** We also used named entity recognition to identify some other named types such as companies, social medias etc.
- **Manual Annotations:** We manually identified and recorded several x labels.

Then we had to **Divide the data into simple and complex** Statista contained both simple and complex chart. We define simple charts as charts with 1 column for y axis.

Complex chart would have more than three column in the data. So based on this criteria charts had to be grouped together.

Finally we had to **Convert to standard format** Finally the data had to be converted into a single standard format.

Knoema

Processing data from knoema was quite simple. Data was first downloaded. Then, the title of the chart and captions were cleaned. While cleaning the title and caption, we removed extra white spaces, newline, normalized the numbers and normalized the entities. /par

That being said, we contacted knoema to ensure we could use their data. They replied that if the source they collected data from was open source, then we could use those data. Thus we had to look into the list data source that knoema use, filter out which sources were open, and filter data accordingly.

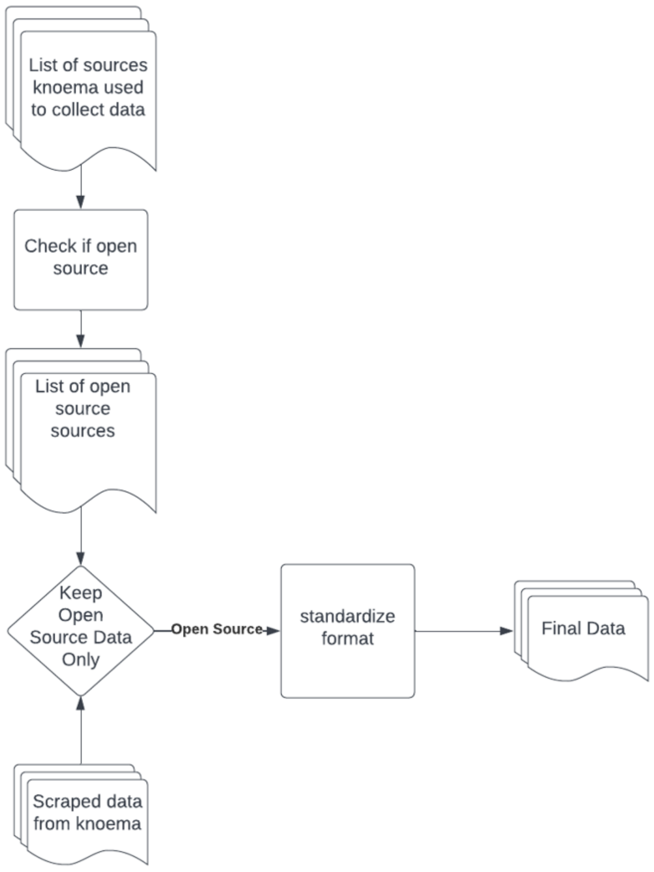

List of sources knoema used to collect data
Check if open source
List of open source sources
Keep Open Source Data Only
Open Source
standardize format
Final Data
Scraped data from knoema

Figure 4.3: Figure showing the general steps involved in preprocessing scraped data from knoema

Table 4.1: Split distribution of ChartSumm

Split	Size
Train	67,488
Valid-e	4,338
Valid-h	4,101
Test-e	4,338
Test-h	4,098
Total	84,363

4.1.3 Test set construction

From our analysis, we have found that summaries from statista contains information beyond the chart information. In table 4.3, we show dataset analysis of ChartSumm.

From the table we can see that on an average, the captions in statista are quite longer compared to the captions in knoema. Also, from our previous analysis we found out that in automatic evaluation, models often fail to perform well because of the topics in reference caption which is out of the scope of the table data. To address this issue, we propose two test sets for our data: Test-e and Test-h. We ensured that in Test-e the reference summaries only contain data from chart data. On the other hand, in Test-h, the captions might contain summaries which might be out of the data. In table 4.1, we show the size of each split of our dataset.

4.1.4 Dataset Analysis

From the table 4.3, we can see that Knoema contains higher number of data cells. The captions of statista is quite large compared to the ones from Knoema.Table 4.2 is the chart distribution based on chart type. Here, we have a large amount of line chart and bar chart but a very small amount of pie chart.

We used Latent Dirichlet allocation (LDA) for topic modeling. After running topic

Table 4.2: Class distribution based on chart type of ChartSumm

Chart Distribution		
Chart type	Statista	Knoema
Linechart	13904	43179
Barchart	26644	0
Piechart	636	0
Total	41184	43179
Sum		84363

Table 4.3: Dataset analysis of ChartSumm

Source	Average Cell Count	Average Summary Length		Average Title Length	
		Token	Chars	Token	Cars
Knoema	55.44	34.76	207.69	8.86	57.18
Statista - simple	13.31	46.96	288.68	9.58	63.08
Statista - complex	37.95	55.54	340.19	10.54	67.08

modeling on our ChartSumm dataset, we found the major topics in our dataset are: economy and politics, society and science, internet and media and public life and health. Figure 4.4 shows the topic distribution of ChartSumm dataset. About 21.6 percent of the samples are economical and political. About 13.03 percent of the samples are from society and science.

4.2 Baseline Models

In order to propose a benchmark dataset we need to propose some baseline models the result of which others will try to beat. We propose four chart data to text models for baseline, and one chart image to text model. We have already explained how these models work in background studies section. In this section we explain how we used those models for out purpose.

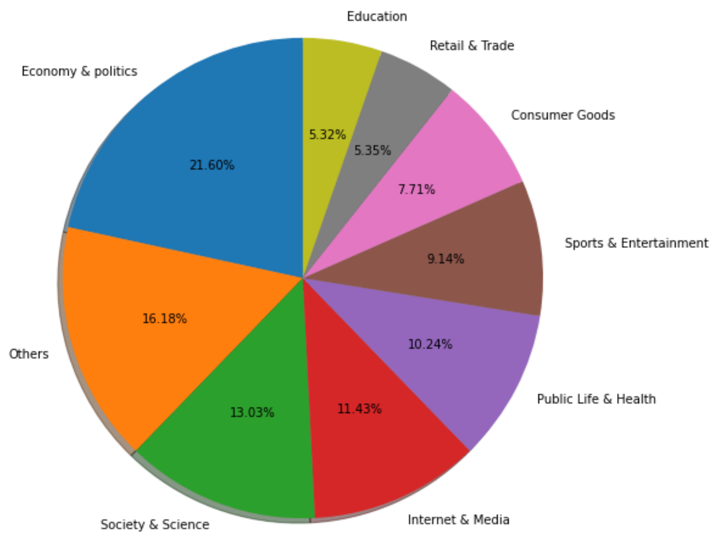


Figure 4.4: Distribution of topic in ChartSumm dataset

4.2.1 Chart2Text

Chart2Text [11] adapted an enhanced version of transformer [42]. For our training we adapted the codebase provided by chart2text repository [2]. We also kept the same hyper parameters used in the paper. The model was trained on google colab using a Tesla P100 GPU.

4.2.2 AutoChart

AutoChart is a template based model for chart summarization. It used Linguist Rhetorical Analysis specified in the paper for generating the description. As this model don't have any training, we just needed to run an inference. For inference, we converted our test data into AutoChart specified format given in their repository [3]. As auto-Chart can not work with complex dataset with more than two column, we only taken first two column from our test dataset while processing our dataset.

4.2.3 T5

T5 is a large pretrained language model trained on multiple sequence to sequence tasks like machine translation, text classification, text summarization, ner etc. For our task we wanted to adapt the pretrained text summarization model. So, we flattened the table by rows and concatenated it with the caption of the table separated by a separator token. After that we trained our model. We used google colab for training the model. We fine-tuned a huggingface implementation of *T5 base model* [4] for 6 epochs and then generated the summary. During fine tuning the maximum input length was 512 tokens and maximum caption length was 128 tokens.

4.2.4 BART

BART is also a large language model pretrained based on denoising auto encoder architecture. It is also based on sequence to sequence model. For training, we flattened the table by rows and concatenated it with the caption of the table separated by a separator token. After that we trained our model. We used google colab for training the model. We fine-tuned a huggingface implementation of *BART Large xsum* [5] for 5 epochs and then generated the summary. During fine tuning the maximum input length was 256 tokens and maximum caption length was 128 tokens.

[2]https://github.com/JasonObeid/Chart2Text
[3]https://gitlab.com/bottle_shop/snlg/chart/autochart
[4]https://huggingface.co/t5-base
[5]https://huggingface.co/facebook/bart-large-xsum

4.2.5 Image Captioning Model

For image captioning we use the standard sequence to sequence like structure, but replace the LSTM encoder with Inception(V3) [43] followed by a fully connected layer. The decoder can then accept the output of the fully connected layer as the context vector for generating the natural language.

The vocabulary only considered words that occurred at least 5 times. For word embedding, we used an embedding size of 256. The number of LSTM layers used was 1. We used *adam* optimizer with a learning rate of 3×10^{-3}. For loss function *CrossEntropyLoss* was used. The model was trained for 6 epochs due to time constraint. Each epochs took over 2.5 hours.

Chapter 5

Evaluation Metric

In order to asses the quality of the dataset, we need to asses if the quality of translations produced by models trained using this dataset. Thus we evaluate the performance of models through various automated evaluation matrices. Human evaluation will be extremely time consuming, and will contain a significant amount of bias.We need numerical scores to perform valid comparisons, but human scoring the quality without any objective mathematical metric will not give a fair score.

5.1 BLEU

BLEU [44] stands for Bilingual Evaluation Understudy. It is used to automatically evaluate machine translated (or generated) text. Generally, BLEU score generates a number between zero to one. It is the measurement of the similarity (n-gram overlaps) between machine generated text and another reference text. Since BLEU is a parameterized score , it mostly depends on the parameters. Preprocessing(tokenization and normalization) of the reference text has some significant effect on BLEU score. Difference user specified preprocessing can frequently change the score. Thus, instead of using BLEU score, it is suggested to use SACREBLEU which expects de-tokenized outputs, applying its own metric-internal preprocessing. It is a python package which downloads and stores references automatically for common test sets. This introduces a 'predictive layer' between references and users. SACREBLEU generates a short version of string that records the parameters. Main features of SACREBLEU are : (i) It automatically downloads common WMT test sets and processes them to plain text. (ii) It produces a short version of string that facilitates cross-paper comparisons. (iii) It properly computes scores on de-tokenized outputs, using WMT standard tokenization. (iv) It produces the same values as the official script used by WMT. (v) It outputs the BLEU score without the comma, so we don't have to remove it. (vi) It supports chrF, chrF++ and Translation error rate metrics and (vii) It performs paired bootstrap re-sampling and paired approximate randomization tests for statistical significance reporting.

Let *Count* be the maximum number of times a candidate n-gram occurs in any single reference translation, *MaxRefCount* be the maximum number of n-grams occurrences in any reference count, *c* be count of words in candidate translation, *r* be count of words in reference translation and *N* be number of N grams.

$$
\begin{aligned}
Count_{clip} &= min(count, MaxRefCount) \\
p_n &= \frac{\sum_{C\in\{candidates\}} \sum_{ngram\in C} Count_{clip}(ngram)}{\sum_{C'\in\{candidates\}} \sum_{ngram'\in C} Count(ngram')} \\
Brevity\ Penalty(BP) &= \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leq r \end{cases} \\
BLEU &= BP.exp(\sum_{n=1}^{N} w_n log(p_n))
\end{aligned}
$$

5.2 BLEURT

BLEURT [45] is an automatic metric that uses machine learning to capture non-trivial semantic similarities across texts. It is trained using a public dataset of ratings (the WMT Metrics Shared Task dataset) as well as user-supplied ratings. It is a reference based evaluation metric for generated English text. It is based on BERT that models human judgements with few training examples. It uses synthetic examples to make a generalized model. It takes two sentences as input: a reference and a candidate, and provides a score that shows how fluent the candidate is and how well it transmits the reference's meaning. It's similar to sentence-BLEU, BERTscore, and COMET scores.

BLEURT is generated by passing reference and generated texts to a trained model. The training objective of the model is as follows:

- input = reference_text, generated_text.

- Output(y) is a human assigned score for the quality of generated text.

- Forward pass input through model to get a predictions($\hat{y}$).

- Using y and $\hat{y}$ calculate $loss$ and back-propagate.

- Train till an acceptable validation accuracy is obtained.

5.3 Perplexity

Perplexity [46] is a metric that measures how effectively a probability model predicts a sample. Perplexity is one technique to evaluate language models in the context of Natural Language Processing. It is the inverse probability of the test set, normalized by the number of words. The equation for calculating perplexity is:

$$PP(W) = \frac{1}{P(w_1, w_2, \ldots, w_n)^{\frac{1}{N}}}$$

Here $P(w_1, w_2, ..., w_n)$ represents the probability of a sentence containing the words $w_1, w_2, ..., w_n$, N represents the order *N-gram*, and $PP(W)$ represents the perplexity of the text W.

In our analysis we used RoBERTa-base [47] Medium Language Generation model for calculating perplexity.

Chapter 6

Result and Analysis

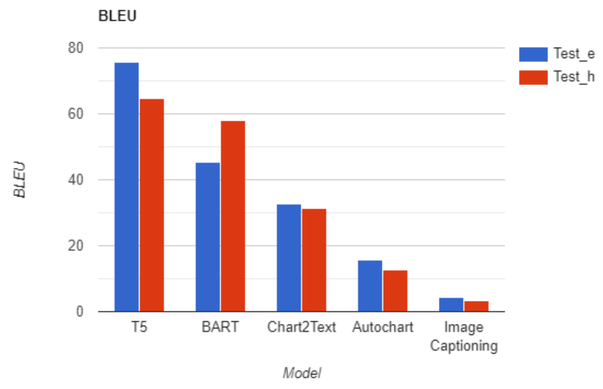
Table 6.1 shows the evaluation results for different models on our ChartSumm dataset. Figures 6.1a,6.1b & 6.1c shows the results in graphical form. From the table and charts, we can see that both T5 has outperformed all other models on both test sets in all three criteria. BART also performed similarly in Test-h along with T5 but performed poorly in Test-e. Although, autochart failed to capture most of the important part from our summary, it generated a well readable and fluent summary as we can see from perplexity score of AutoChart. Chart2Text failed to perform well on our dataset as it was trained on vanilla transformer architecture. A longer training time can give a better result on this model.

The image captioning model performed quite poor compared to the other model. It can be intuitively said as the model was fed only the image as input, it failed to capture the important information from the chart. Also the quality of the generation from image captioning model was also poor.

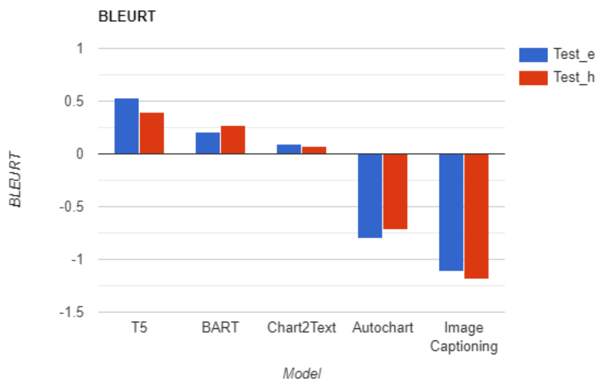
Figure 6.2 and 6.3 shows an example of generated summaries by two of our best performing models. In both of the cases we can see that both T5 and BART could generate the key portion of the reference summaries. It could also generate fluent and cohesive summaries.

Table 6.1: Evaluation results for different models on our dataset. ↑ : means higher is better, ↓ means lower is better.

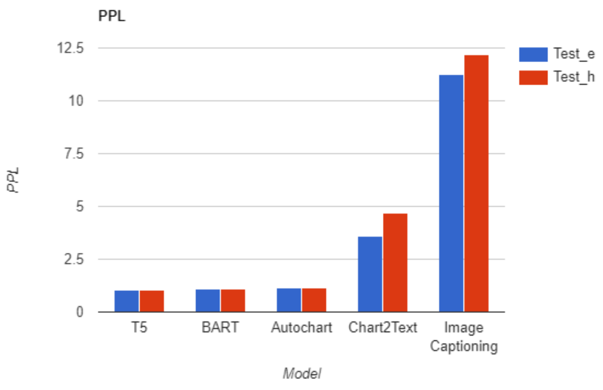
Models	BLEU ↑		BLEURT ↑		PPL ↓	
	Test-e	Test-h	Test-e	Test-h	Test-e	Test-h
AutoChart	15.70	12.70	-0.80	-0.72	1.16	1.16
Chart2Text	32.67	31.21	0.09	0.07	3.60	4.68
T5	75.72	64.78	0.53	0.40	1.06	1.03
BART	45.47	58.03	0.21	0.27	1.10	1.09
Image Captioning	4.36	3.44	-1.11	-1.19	11.23	12.19



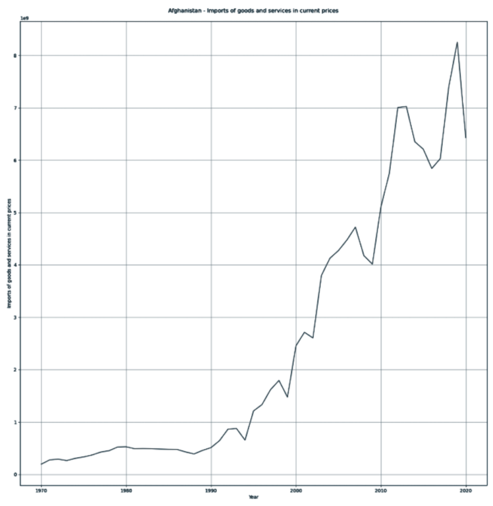
(a) Figure showing BLEU score of each models. Higher is better.



(b) Figure showing BLEURT score of each models. Higher is better.



(c) Figure showing PPL score of each models. Lower is better.

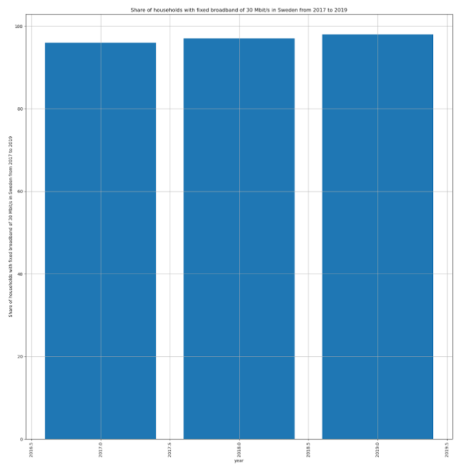


Reference:
In 2020, imports of goods, services and primary income for Afghanistan was 7,034 million US dollars. Though Afghanistan imports of goods, services and primary income fluctuated substantially in recent years, it tended to increase through 1983 - 2020 period ending at 7,034 million US dollars in 2020.

T5:
In 2020, the imports of goods, services and primary income for Afghanistan was 7,034 million US dollars. Though it substantially fluctuated in recent years, it tended to increase through 1983-2020 period.

BART:
In 2020, Afghanistan's imports of goods, services and primary income amounted to 7,034 million US dollars. Though Afghanistan's imports fluctuated substantially in recent years, it tended to increase through 1983 - 2020 period.

Figure 6.2: Generated summaries by T5 and BART on one of the example from test-e. The blue marked lines shows exact similar lines have been generated as the reference line. Also, the red markers show the important aspects of the summary which was captured by both of the models.



Reference:
This statistic shows the share of households in Sweden with access to fixed broadband connection of 30 Mbit/s, from 2017 to 2019. In 2017, the rate was at 96 percent, and it increased to 98 percent in 2019.

T5:
This statistic shows the share of households in Sweden with access to fixed broadband connection of 30 Mbit/s, from 2017 to 2019. The share of households with access to fixed broadband connection increased from 96 percent in 2017 to 98 percent in 2019.

BART:
This statistic shows the share of households in Sweden with access to fixed broadband connection of 30 Mbit/s, from 2017 to 2019. In 2019, the share of households with access to fixed broadband connection increased from 96 percent in 2017 to 98 percent in 2019.

Figure 6.3: Generated summaries by T5 and BART on one of the example from test-h. The blue marked lines shows exact similar lines have been generated as the reference line. Also, the red markers show the important aspects of the summary which was captured by both of the models.

Chapter 7

Conclusion

In our research, we propose a new large scale dataset for chart to text summarization that can used to train models for chart summarization task. The generated summaries then can be used for various tasks like improving accessibility for visually impaired people, improving performance of information retrieval systems. Our evaluation shows that both T5 and BART can successfully generate precised summaries of a given chart after being trained on our dataset achieving a decent BLEU, BLEURT and perplexity score. However, different studies shows that automatic evaluation often fails to capture factual correctness, cohesiveness and fluency of the generated text. Extensive human evaluation can be done to test these criteria to have a better understanding on the baseline model's performance. In future, we would like to run extensive human evaluation to have a better understanding. In addition we would also like to experiment with some new architectures that could perform better in chart to text summarization task. We hope that *ChartSumm* will be able to serve as a useful benchmark for models and motivate other researchers to explore this new low resourced research domain.

REFERENCES

[1] L. Gong, J. Crego, and J. Senellart, "Enhanced transformer model for data-to-text generation," in *Proceedings of the 3rd Workshop on Neural Generation and Translation*, Nov. 2019.

[2] K. Chen, F. Li, B. Hu, W. Peng, Q. Chen, H. Yu, and Y. Xiang, "Neural data-to-text generation with dynamic content planning," *Knowledge-Based Systems*, 2021.

[3] J. Sreevalsan-Nair, K. Dadhich, and S. C. Daggubati, "Tensor fields for data extraction from chart images: bar charts and scatter plots," in *Topological Methods in Data Analysis and Visualization VI.* Springer, 2021.

[4] J. Luo, Z. Li, J. Wang, and C.-Y. Lin, "Chartocr: Data extraction from charts images via a deep hybrid framework," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[5] F. Zhou, Y. Zhao, W. Chen, Y. Tan, Y. Xu, Y. Chen, C. Liu, and Y. Zhao, "Reverse-engineering bar charts using neural networks," *Journal of Visualization*, vol. 24, no. 2, 2021.

[6] K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "Barchartanalyzer: Digitizing images of bar charts." in *BarChartAnalyzer*, 2021.

[7] J. Obeid and E. Hoque, "Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model," *arXiv preprint arXiv:2010.09142*, 2020.

[8] J. Zhu, J. Ran, R. K.-w. Lee, K. Choo, and Z. Li, "Autochart: A dataset for chart-to-text generation task," *arXiv preprint arXiv:2108.06897*, 2021.

[9] T.-Y. Hsu, C. L. Giles, and T.-H. Huang, "SciCap: Generating captions for scientific figures," in *Findings of the Association for Computational Linguistics: EMNLP 2021.* Punta Cana, Dominican Republic: Association for

Computational Linguistics, Nov. 2021, pp. 3258–3264. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.277

[10] S. Kanthara, R. T. K. Leong, X. Lin, A. Masry, M. Thakkar, E. Hoque, and S. Joty, "Chart-to-text: A large-scale benchmark for chart summarization," *arXiv preprint arXiv:2203.06486*, 2022.

[11] J. Obeid and E. Hoque, "Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model," in *Proceedings of the 13th International Conference on Natural Language Generation*. Dublin, Ireland: Association for Computational Linguistics, Dec. 2020, pp. 138–147. [Online]. Available: https://aclanthology.org/2020.inlg-1.20

[12] D. H. Kim, E. Hoque, and M. Agrawala, *Answering Questions about Charts and Generating Visual Explanations*. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–13. [Online]. Available: https://doi.org/10.1145/3313831.3376467

[13] G. Carenini, C. Conati, E. Hoque, and B. Steichen, "User task adaptation in multimedia presentations." in *UMAP Workshops*. Citeseer, 2013.

[14] L. Ferres, G. Lindgaard, L. Sumegi, and B. Tsuji, "Evaluating a tool for improving accessibility to charts and graphs," *ACM Trans. Comput.-Hum. Interact.*, vol. 20, no. 5, nov 2013. [Online]. Available: https://doi.org/10.1145/2533682.2533683

[15] S. Carberry, S. Elzer, and S. Demir, "Information graphics: an untapped resource for digital libraries," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006.

[16] Z. Li, M. Stagitis, S. Carberry, and K. F. McCoy, "Towards retrieving relevant information graphics," in *Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval*, 2013, pp. 789–792.

[17] E. Reiter, S. G. Sripada, J. Hunter, J. Yu, and I. P. Davy, "Choosing words in computer-generated weather forecasts," *Artif. Intell.*, vol. 167, pp. 137–169, 2005.

[18] M. Fasciano and G. Lapalme, "Intentions in the coordinated generation of graphics and text from tabular data," *Knowledge and Information Systems*, vol. 2, no. 3, pp. 310–339, 2000.

[19] V. O. Mittal, J. D. Moore, G. Carenini, and S. Roth, "Describing complex charts in natural language: A caption generation system," *Computational*

Linguistics, vol. 24, no. 3, pp. 431–467, 1998. [Online]. Available: https://aclanthology.org/J98-3004

[20] "Quillbot," https://quillbot.com/. [Online]. Available: https://quillbot.com/

[21] "Wordsmith," https://lexically.net/wordsmith/index.html. [Online]. Available: https://lexically.net/wordsmith/index.html

[22] H. Mei, M. Bansal, and M. R. Walter, "What to talk about and how? selective generation using lstms with coarse-to-fine alignment," *arXiv preprint arXiv:1509.00838*, 2015.

[23] J. Zhu, J. Ran, R. K.-W. Lee, Z. Li, and K. Choo, "AutoChart: A dataset for chart-to-text generation task," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. Held Online: INCOMA Ltd., Sep. 2021, pp. 1636–1644. [Online]. Available: https://aclanthology.org/2021.ranlp-main.183

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

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

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

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, 2000.

[29] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

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

[31] 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.

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

[33] S. Wiseman, S. M. Shieber, and A. M. Rush, “Challenges in data-to-document generation,” *arXiv preprint arXiv:1707.08052*, 2017.

[34] R. Puduppully, L. Dong, and M. Lapata, “Data-to-text generation with entity modeling,” *arXiv preprint arXiv:1906.03221*, 2019.

[35] D. Bahdanau, D. Serdyuk, P. Brakel, N. R. Ke, J. Chorowski, A. Courville, and Y. Bengio, “Task loss estimation for sequence prediction,” *arXiv preprint arXiv:1511.06456*, 2015.

[36] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, “Pointing the unknown words,” *arXiv preprint arXiv:1603.08148*, 2016.

[37] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[38] R. Lebret, D. Grangier, and M. Auli, “Neural text generation from structured data with application to the biography domain,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1203–1213. [Online]. Available: https://aclanthology.org/D16-1128

[39] C. Thomson, E. Reiter, and S. Sripada, “SportSett:basketball - a robust and maintainable data-set for natural language generation,” in *Proceedings of the Workshop on Intelligent Information Processing and Natural Language Generation*. Santiago de Compostela, Spain: Association for Computational Lingustics, Sep. 2020, pp. 32–40. [Online]. Available: https://aclanthology.org/2020.intellang-1.4

[40] “knoema,” https://knoema.com/atlas. [Online]. Available: https://knoema.com/atlas

[41] “statista,” https://www.statista.com/. [Online]. Available: https://www.statista.com/

[42] L. Gong, J. M. Crego, and J. Senellart, “Enhanced transformer model for data-to-text generation,” in *Proceedings of the 3rd Workshop on Neural Generation and Translation*, 2019, pp. 148–156.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[44] M. Post, "A call for clarity in reporting BLEU scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 186–191. [Online]. Available: https://aclanthology.org/W18-6319

[45] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7881–7892. [Online]. Available: https://aclanthology.org/2020.acl-main.704

[46] L. Azzopardi, M. Girolami, and K. van Risjbergen, "Investigating the relationship between language model perplexity and ir precision-recall measures," ser. SIGIR '03. New York, NY, USA: Association for Computing Machinery, 2003. [Online]. Available: https://doi.org/10.1145/860435.860505

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