

**Encode-Attend-Refine-Decode:  
Enriching Contextual Representations for Natural  
Language Generation**

*A Thesis*

*submitted by*

**PREKSHA NEMA**

*for the Award of the Degree*

*of*

**DOCTOR OF PHILOSOPHY**



**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY MADRAS**

**April 2021**

# THESIS CERTIFICATE

This is to certify that the thesis titled **Encode-Attend-Refine-Decode: Enriching Contextual Representations for Natural Language Generation**, submitted by **Preksha Nema (CS15D201)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Doctor of Philosophy**, is a bonafide record of the research work done by her under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

**Mitesh M. Khapra**

Research Guide

Associate Professor

Department of Computer Science and Engineering

IIT-Madras, 600 036

**Balaraman Ravindran**

Research Co-Guide

Professor

Department of Computer Science and Engineering

IIT-Madras, 600 036

Place: Chennai

Date: 1-05-2021

## **ACKNOWLEDGEMENTS**

My Ph.D. journey, like any other, has been filled with ups and downs. Many roadblocks in between kept on testing my perseverance and patience, but because of the continuous support from my advisors, colleagues, friends, and family, I am writing these last two pages of my Ph.D. thesis.

Little did I know back then, while doing M.Tech, that I will be doing a Ph.D. But thanks to my guide Dr. Mitesh M. Khapra, to whom I will always be grateful for making me realize that I had the potential of pursuing the same. He helped me at various stages of this journey and paved the way whenever I was deviating. He always had my best interest at heart, be it at the very early stage, where he helped me with my Google Ph.D. Fellowship application, which has been a massive boost for my Ph.D., or at the very end stages where he was very supportive at such uncertain times and patiently helped me shape my thesis. Working with him in several research works and TAing for his courses, I learned how to ask the right questions while pursuing a project. His attention to detail and passion while teaching a concept, presenting some ideas, or writing papers are inspirational. I will always be grateful to him for helping, inspiring, and supporting me to pass this crucial milestone of my life.

I am very thankful for the opportunity that Prof. Balaraman Ravindran was my co-advisor. He was very supportive throughout my Ph.D. The discussion with him and his inquisitive questions always enriched our research projects. During my initial phase of research, he encouraged me to take up my MTP on abstractive summarization and always connected me to the right set of scholars who eventually helped in understanding the basics of deep learning. When Prof. Mitesh joined IITM, Prof. Ravi introduced me to him to collaborate for a similar project on summarization, which eventually turned out to be my first paper. I have learnt from him that it is equally important to be proud of your work and increase visibility, as it is to work on exciting ideas, to be a successful researcher.

I am also thankful to Dr. Hema A. Murthy, Dr. Sutanu Chakraborti, and Dr.C.Chandra Sekhar for being so passionate about their courses and introduced me to interesting concepts in Machine Learning and NLP, which eventually formed the basis of my thesis. I am also thankful to Dr. P. Sreenivasa Kumar, Dr. N.S. Narayanaswamy, Dr. Kaushik Mitra, and Dr. Sutanu Chakraborti for serving on my Doctoral Committee and providing helpful feedback in my Doctoral Committee meetings.

I was fortunate to have research collaborations with Anirban Laha, Parag Jain and Dr. Balaji Vasani Srinivasan, and Dr. Pratyush Kumar Panda. Several fruitful discussions helped in witnessing how the same problem can be viewed from different angles.

Working on my research projects would not have been so thrilling and exciting if it was not for my research collaborators: Shreyas, Soham, Ananya, Madhura, Aakriti, Sahana, Akash, Deep, and Sharan. Thank you all for sharing the same passion and curiosity on getting things to work, which made the numerous night-outs and endless discussions worth it.

I would also like to thank Annie Louis, Alexandros Karatzoglou, Filip Radlinski, and Pauline Anthonysamy for being such great mentors during my Google Internships and helping me get through the initial Covid lockdown period in London.

I am grateful to Ayushi, Sai, Manisha, Sonam, Disha, Abhirut, Mayuresh, Abhinay, and Saksham for all their love and support. During these six years, I have been aloof and not been around most of the time, but you still were there to lend me your ear whenever I was overwhelmed or anxious.

This journey would not have been so memorable if it wasn't for Sakshee, Rimpa, Madhura, Snehal, KV, Naman, and Siddharth. I was scared and skeptical when I had joined the M.Tech program after having worked for three years, but you guys made this transition very easy. In my next phase, The mundane routine of working at the lab became enjoyable with Gargi, PK, Rahul, Arjun, Vishvesh, Vinod, Nikita, Tarun, Suman, Nitesh, Pritha, Jaya, and Shubham. I thank you all for the numerous chai sessions in the department and IRCTC. Also, I absolutely cherished sharing our shared love for food with Madhura (M.Tech), Priyesh, Rahul, Madhura (M.S.), Aakriti, Vishvesh, Vinod, Anmol, and Gargi.

I will always be indebted to my parents for being so encouraging during my Ph.D. I always have and will look up to my father for being so rigorous and passionate about even a small task that he does, and to my mother for teaching me to take things one step at a time and to always smile and be patient in dealing with anything that comes our way. I have always tried to learn a little of both qualities from them. I think that had played a significant role in successfully achieving this milestone. I thank my younger brother Tejaswi who always motivates me to sit back and relax and enjoy and always cherish what has been achieved.



# ABSTRACT

Natural Language Generation (NLG) refers to the process of automatically generating human-understandable text from a given context. It includes a wide variety of tasks such as machine translation, abstractive summarization, automatic question generation, structured data to text generation, *etc.* In the past decade, the field of NLG has rapidly evolved from statistical models to large scale transformer based models. In this thesis, we focus on Recurrent Neural Network (RNN) based NLG models, popularly known as sequence-to-sequence models (seq2seq). These models are also known as *encode-attend-decode* models as they typically contain three modules, *viz.*, (i) encoder (ii) attention network and (iii) decoder. The encoder computes a contextual representation for the tokens in the input sequence. The attention network computes a weighted sum of the token representations, giving more weight to important tokens. The decoder uses this weighted representation to generate the output, one token at a time.

While these models are very popular, several studies have highlighted their limitations. In this work, we focus on three such limitations: (i) the inability to avoid repeating phrases in the output, (ii) the inability to track previously covered fields in the input sequence, and (iii) the inability to use task-specific rewards, which improve generation quality. The first limitation is *task-agnostic* and has been reported for several tasks such as machine translation, dialogue generation, abstractive summarisation, *etc.* In particular, it has been observed that seq2seq models generate non-fluent output containing repeating phrases such as “*China Germany Germany Germany at world youth championship*”. To alleviate this problem, we propose a *refine* module which ensures that the context vectors fed to the decoder at each time step are orthogonal to the context vectors at previous time steps. We also introduce a learnable soft-orthogonalization parameter,  $\gamma$  which ensures that the context vectors are not orthogonalized when it is not desirable to do so (*e.g.*, for generating an output such as “*He kept talking and talking and talking.*”, which is natural in certain contexts). We evaluate the proposed refine-

ment on the task of query-based abstractive summarization and show quantitative gains over the existing state-of-the-art models.

The second limitation is *task-specific* and is observed in the context of generating descriptions from structured data, *i.e.*, a table of facts containing information organized in different fields (rows). Existing seq2seq models are too generic and hence fail to capture certain task-specific requirements. For example, while writing a description of a table a human would continue attending to a field for a few timesteps till all the information from that field has been rendered and then never return back to this field (because there is nothing left to say about it). To capture this behavior, we use a gated orthogonalization mechanism to ensure that a field is remembered for a few time steps and then forgotten. In effect, we refine the representations computed by the attention network before feeding them to the decoder. We evaluate our model on a publicly available dataset and show that the proposed refinement improves the quality of the generated description.

The third limitation arises from the inability of seq2seq models to use task-specific rewards. For example, consider the Automatic Question Generation (AQG) task where given a passage and an answer, the task is to generate the corresponding question. It is desired that the generated question should be (i) grammatically correct, (ii) answerable from the passage, and (iii) specific to the given answer. An analysis of existing seq2seq models showed that they produce questions that do not adhere to one or more of the aforementioned qualities. To capture the desired properties, we first propose a new metric for AQG, which gives weightage to *answerability* in addition to fluency. We then propose a method that tries to mimic the human process of generating questions by first creating an initial draft and then refining it based on the reward/penalty given by the above metric. We evaluate our model on multiple datasets and show that it outperforms existing state-of-the-art methods.

A common theme across all the solutions proposed in this work for the three limitations described above is a *refine* module. The specifics of the refine module depend on the task and limitation being addressed. We believe that it can also be extended to other tasks and the recent transformer based models giving rise to a new paradigm, *viz.*, *encode-attend-refine-decode* as opposed to *encode-attend-decode*.



# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>i</b>
<b>ABSTRACT</b>	<b>v</b>
<b>LIST OF TABLES</b>	<b>xv</b>
<b>LIST OF FIGURES</b>	<b>xix</b>
<b>NOTATION</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Limitations of Sequence to Sequence Models . . . . .	3
1.2 Objectives of the Thesis . . . . .	6
1.3 Contributions of this Thesis . . . . .	7
1.4 Outline of the Thesis . . . . .	9
<b>2 Background</b>	<b>11</b>
2.1 Relevant NLG Tasks . . . . .	11
2.1.1 Abstractive Summarization . . . . .	11
2.1.2 Query-Based Abstractive Summarization . . . . .	11
2.1.3 Structured Data to Text Generation . . . . .	13
2.1.4 Question Generation . . . . .	13
2.2 An overview of Neural Networks . . . . .	14
2.2.1 Artificial Neuron . . . . .	14
2.2.2 Artificial Neural Networks . . . . .	15
2.2.3 Training ANNs . . . . .	16
2.3 Recurrent Neural Networks . . . . .	17
2.3.1 Learning Algorithms for RNNs . . . . .	18
2.4 Long Short Term Memory . . . . .	19

2.5	Sequence-to-Sequence Models . . . . .	21
2.5.1	Encoder Module . . . . .	21
2.5.2	Decoder Module . . . . .	22
2.6	Attention Mechanisms . . . . .	23
2.7	Evaluation Metrics . . . . .	25
2.7.1	BLEU . . . . .	25
2.7.2	NIST . . . . .	26
2.7.3	METEOR . . . . .	27
2.7.4	ROUGE . . . . .	28
2.8	Summary . . . . .	29
<b>3</b>	<b>Related Work</b>	<b>30</b>
3.1	State of the art models for different tasks considered in this thesis . .	32
3.1.1	Task: Structured data-to-text generation . . . . .	32
3.1.2	Automatic Question Generation . . . . .	37
3.1.3	Query Based Abstractive Summarization . . . . .	43
3.2	Works addressing the same limitations in Seq2Seq models as identified in this thesis . . . . .	49
3.2.1	Avoiding Repeating Phrases . . . . .	49
3.2.2	Better Transition among fields for Structured data to Text Gen- eration . . . . .	54
3.2.3	Improving answerability for Question Generation . . . . .	55
3.2.4	Other Limitations . . . . .	56
3.3	Works using similar techniques as proposed in this thesis . . . . .	58
3.3.1	Using Orthogonalization to Diversify Context Vectors . . . . .	58
3.3.2	Using explicit reward signals for improving NLG . . . . .	59
3.3.3	Generating multiple drafts . . . . .	60
3.4	Summary . . . . .	61
<b>4</b>	<b>Avoiding Repeating Phrases in NLG</b>	<b>62</b>
4.1	Introduction . . . . .	62
4.2	Model Architecture . . . . .	65
4.2.1	Encode . . . . .	66

4.2.2	Attend . . . . .	66
4.2.3	Refine . . . . .	68
4.2.4	Decode . . . . .	70
4.3	Baseline Methods . . . . .	70
4.4	Query Based Abstractive Summarization Dataset . . . . .	71
4.5	Experimental Setup . . . . .	73
4.5.1	Implementation Details . . . . .	75
4.6	Results and Discussions . . . . .	77
4.7	Summary . . . . .	79
<b>5</b>	<b>Exploiting Task Specific Characteristics for improving Adequacy</b>	<b>80</b>
5.1	Introduction . . . . .	80
5.2	Proposed model . . . . .	83
5.2.1	Encode . . . . .	84
5.2.2	Attend . . . . .	86
5.2.3	Refine . . . . .	87
5.2.4	Decode . . . . .	89
5.2.5	Copying Mechanism . . . . .	89
5.3	Experimental setup . . . . .	90
5.3.1	Datasets . . . . .	90
5.3.2	Models compared . . . . .	91
5.3.3	Hyperparameter tuning . . . . .	91
5.4	Results and Discussions . . . . .	92
5.4.1	Comparison of different models . . . . .	92
5.4.2	Human Evaluation . . . . .	94
5.4.3	Performance on different languages . . . . .	94
5.4.4	Visualizing Attention Weights . . . . .	95
5.4.5	Out of domain results . . . . .	98
5.5	Summary . . . . .	101
<b>6</b>	<b>Designing Task Specific Metric for improving Answerability</b>	<b>102</b>
6.1	Introduction . . . . .	102

6.1.1	Q-Metric: A better metric for AQG . . . . .	103
6.1.2	Using Q-Metric to improve AQG systems . . . . .	105
6.2	Human Judgments For Answerability . . . . .	106
6.2.1	Creating Noisy Questions . . . . .	107
6.2.2	Instructions provided to human annotators . . . . .	109
6.2.3	Human-Human Correlation . . . . .	110
6.2.4	Correlation between human scores and existing evaluation metrics . . . . .	112
6.3	Modifying existing metrics for AQG . . . . .	113
6.3.1	Tuning the weights $w_i$ 's and $\delta$ . . . . .	114
6.3.2	Correlation between Human scores and different Q-METRICS . . . . .	115
6.3.3	Qualitative Analysis . . . . .	116
6.4	Extrinsic evaluation . . . . .	116
6.5	Encode-Attend-Refine-Decode Model for AQG . . . . .	120
6.5.1	Encode Module . . . . .	120
6.5.2	Attend Module . . . . .	122
6.5.3	Decode Module . . . . .	124
6.5.4	Refine Module . . . . .	124
6.5.5	Training Objective . . . . .	125
6.6	Experimental Details . . . . .	126
6.6.1	Datasets for AQG . . . . .	126
6.6.2	Implementation Details . . . . .	126
6.7	Results and Discussions . . . . .	127
6.7.1	EARD's performance across datasets . . . . .	127
6.7.2	Human Evaluations . . . . .	128
6.7.3	Analysis of Refined Draft and Initial Draft . . . . .	129
6.7.4	Analysis of Reward Based Training Objective . . . . .	131
6.7.5	Impact of character and positional embeddings . . . . .	133
6.7.6	Case Study: Originality of the Questions . . . . .	134
6.8	Summary . . . . .	135

**7 Conclusion and Future Work 137**

7.1 Future Directions . . . . .	139
<b>LIST OF PAPERS BASED ON THESIS</b>	<b>161</b>



## LIST OF TABLES

1.1	Samples of erroneous output produced by encode-attend-decode models for various NLG tasks. The generated outputs have been taken from the respective mentioned works. * refers to the encode-attend-decode models taken as baselines in this thesis. . . . .	4
2.1	Example of a document and different possible manually generated summaries for the given document. . . . .	12
2.2	Example of a summary for document in Table 2.1 which is specific to the given query. . . . .	12
2.3	Example of a Wikipedia Infobox (structured data) and different possible descriptions of the Infobox. . . . .	13
2.4	Example of a document and different questions that can be generated from it. These examples are taken from SQuAD dataset (Rajpurkar <i>et al.</i> , 2016). . . . .	14
2.5	Example of a question corresponding to the given (document, answer) pair from SQuAD dataset (Rajpurkar <i>et al.</i> , 2016). . . . .	14
3.1	Existing RNN-based seq2seq models for question generation given a text. Such models broadly consist of answer-aware module, and other features such as <b>CM</b> : Copying Mechanism, <b>PL</b> : Paragraph Level Information, <b>LF</b> : Linguistic Features. The last column enumerates the datasets used to validate the proposed model. . . . .	41
4.1	Examples showing repeated words in the output of encoder-decoder models. . . . .	65
4.2	Average length of passages/queries/summaries in the dataset. . . . .	73
4.3	Performance of various models using full-length ROUGE metrics. . . . .	75
4.4	Summaries generated by different models. In general, we observed that the baseline models which do not use a diversity based attention model tend to produce more repetitions. Notice that the last example shows that our model is not very aggressive in dealing with the history and is able to produce valid repetitions (treated ... treated) when needed. . . . .	76
4.5	Generated Samples where our best model, <b>EARD-SD<sub>2</sub></b> ;, does not restore the semantic meaning or does not generate the summary specific to the given query. . . . .	77

4.6	The number of sentences with repeated unigrams across generated summaries from various models. We average out the total number of sentences across 10 folds. We can infer that the model with hard orthogonalization in Diversity LSTM generates summaries with least number of repetitions. . . . .	78
5.1	Examples of generated descriptions from baseline model. The first example is incoherent. In the second example, the model hops between “Name” and “Governor-General” fields. For the last example, the <i>nationality</i> is incorrect generated by a baseline model is incorrect.	82
5.2	Dataset Statistics for Wikipedia Infoboxes for generating descriptions in various languages. Note that the structured data is also present in the corresponding language. . . . .	90
5.3	Comparison of different models on the English WIKIBIO dataset. . . . .	92
5.4	Examples of generated descriptions from different models. For the last two examples, <i>name</i> generated by the basic seq2seq model is incorrect because it attended to the <i>preceded by</i> field. . . . .	93
5.5	Qualitative Comparison of Model A (Seq2Seq) and Model B (our model).	94
5.6	Comparison of different models on the French WIKIBIO dataset. . . . .	95
5.7	Comparison of different models on the German WIKIBIO dataset. . . . .	95
5.8	In this table, we focus on the shift in the performance when the model does not see any training samples from the corresponding domain, to samples being added gradually. We report the BLEU-4 scores for Wikipedia Infoboxes from two domains: Arts and Sports. We can infer that without the target domain samples the performance drops significantly. However, the performance recovers quickly through fine-tuning the model on 5K samples. . . . .	98
6.1	Samples of generated questions from Baseline (EAD), Encode-Attend-Refine-Decode (EARD), and Reward-EARD model on the SQuAD dataset. Answers are shown in <a href="#">blue</a> . . . . .	105
6.2	Some examples to highlight that the generated questions from Baseline (EAD) models are difficult to be analyzed systematically. . . . .	107
6.3	Instructions along with the examples. The words that are strike out were removed from the original question to create a noisy question. . . . .	109
6.4	Interpretation for the inter-annotator agreement score using Cohen’s $\kappa$ .	111
6.5	Inter annotator agreement, Pearson and Spearman coefficients between Human Scores. . . . .	111
6.6	Correlation between existing metrics and human judgments. The values with * are <b>not</b> statistically significant (p-value > 0.01). . . . .	112



6.7	Coefficients learnt for $Q$ -BLEU1 from human judgments across different datasets. . . . .	115
6.8	Correlation between proposed Q-Metric and human judgments. All the correlations have a p-value $< 0.01$ and hence statistically significant. . . . .	116
6.9	Human (Gold) and $Q$ -Metric scores for some of the examples from the collected human-evaluation data. . . . .	117
6.10	Performance obtained by training on different types of noisy questions (WikiMovies). . . . .	118
6.11	Performance obtained by training on different types of noisy questions (SQuAD). . . . .	118
6.12	Performance obtained by training on different types of noisy questions (VQA). . . . .	118
6.13	Comparison of EARD model with existing approaches and EAD model. Here * denotes our implementation of the corresponding work. . . . .	128
6.14	An example where <b>EAD</b> model was better than <b>EARD</b> . The ground truth answers are shown in blue. . . . .	129
6.15	Comparison between Preliminary and Refinement decoders in <b>EARD</b> Model for SQuAD Sentence Level QG. . . . .	129
6.16	Generated samples by Preliminary and Refinement Decoders in our proposed <b>EARD</b> model. We observe that the Refinement Decoder improves the draft by adding relevant phrases " <i>{causes cervical cancer, terrible headaches}</i> ". . . . .	130
6.17	Generated samples by Preliminary and Refinement Decoders in our proposed <b>EARD</b> model. We observe that the Refinement Decoder improves the draft by correcting question types/function words in the initial draft. . . . .	132
6.18	Impact of Reward- <b>EARD</b> on various datasets when fluency and answerability are used as reward signals. . . . .	133
6.19	An example of question with significant overlap with the passage. The answer is shown in blue. . . . .	133
6.20	An example where Reward-EARD(Originality) is better than EARD. . . . .	134



## LIST OF FIGURES

1.1	A basic seq2seq model consisting of encode-attend-decode modules. The encoder is responsible for learning meaningful contextual representations for words in the input. The decoder then predicts one word at a time by <i>attending to</i> relevant words in the input using the attend module. . . . .	2
1.2	Seq2seq architecture with proposed refine module. The refine module enriches the attended input representations computed from attend module and then passes it to the decode module. . . . .	6
2.1	An artificial neuron (left) and a biological neuron (right). . . . .	14
2.2	A simple feed-forward neural network with two hidden layers. . . . .	15
2.3	Backward flow of error for weight $w$ for a sample feedforward network with two hidden layers. . . . .	17
2.4	Recurrent Neural Network . . . . .	17
2.5	The computations in one timestep of a Long Short Term Memory Cell. . . . .	20
2.6	An RNN-based sequence to sequence model. It consists of encode, attend, and decode modules. . . . .	24
3.1	An overview of the existing works. across three dimensions: <b>Tasks</b> : QBAS (Query based Abstractive Summarization); AQG (Automatic Question Generation); SD2T (Structured Data to Text Generation); <b>Limitations</b> : Avoiding Repeating Phrases, Better Transitions among fields, Improving Answerability; <b>Strategies</b> . We highlight the works based on this thesis in <b>red</b> . Note that the enlisted works are specific to RNN-based seq2seq models. . . . .	31
3.2	Taxonomy of models proposed for the task of Structured Data to Text Generation . . . . .	32
3.3	Taxonomy of models proposed for the task of Automatic Question Generation . . . . .	38
3.4	Taxonomy of models proposed for the task of Query Based Abstractive Summarization . . . . .	43
3.5	Taxonomy for works that avoid repeating phrases in the output . . . . .	50

4.1	Our proposed model is based on encode-attend-refine-decode paradigm. The attend module generates a context vector $c_t$ , by attending on to the passage word representations based on the decoder’s hidden state $s_{t-1}$ . This context vector $c_t$ , is then passed to the refine module, which diversifies $c_t$ with respect to the history of the context vectors $d'_{t-1}$ stored in the refine module. The diversified context vector $d_t$ is then passed to the decode module to generate the next word. . . . .	63
4.2	Queries associated with the topic “algae biofuel”. . . . .	72
4.3	Passages and summaries for a given query. . . . .	72
5.1	The goal is to generate the following description when this infobox is passed as an input: <i>Karl Dykhuis born (born July 8, 1972) is a Canadian former professional ice hockey defenceman who played 12 seasons in the National Hockey League (NHL) for the Chicago Blackhawks, Philadelphia Flyers, Tampa Bay Lightning and Montreal Canadiens.</i>	81
5.2	Our proposed model consists of i) a hierarchical encoder that encodes the infobox at the field and value level, ii) attend module with three sub-components: micro, macro, and fused attention networks, iii) refine module models the <i>stay-on</i> and <i>never-look-back</i> characteristics. Here $d_t$ represents the history of filed level context vectors, iv) a decoder that generates the description conditioned on value-level and refined field-level context vectors. . . . .	84
5.3	Comparison of the attention weights and descriptions produced for Infobox in Figure 5.3a. . . . .	96
5.4	Wikipedia Infobox for Matthias Hagner. . . . .	96
5.5	Comparison of the attention weights and descriptions in German produced for Infobox in Figure 5.4. . . . .	97
5.6	Comparison of the attention weights and descriptions (see highlighted boxes) produced by an out-of-domain model with and without fine tuning for the Infobox in Figure 5.6a . . . . .	99
5.7	Wikipedia infobox for Sheppard Dillon. . . . .	100
5.8	Comparison of the attention weights and descriptions (see highlighted boxes) produced by an out-of-domain model with and without fine-tuning for the Wikipedia infobox in Figure 5.7. . . . .	100
6.1	We plot the ratings given by Annotator A and B across different datasets. From the above plots, we can infer that the ratings are linearly correlated. . . . .	111

6.2	Our Proposed Model with encode-attend-refine-decode paradigm. The decode module provides an initial draft of the question to the refine module. The refine module then improves this initial draft by attending to the initial draft and the passage-answer representation simultaneously. . . . .	119
6.3	Generated Question Length Distribution for Preliminary and Refinement Decoders. . . . .	131
6.4	Attention plots for a) E2P, b) E2R, c) P2R respectively <b>Initial Draft:</b> “What is the name of the oncogenic virus?” <b>Refined Draft:</b> “What is the name of the virus that causes cervical cancer?” . . . . .	132
6.5	Originality Analysis: Plot of Q-BLEU score vs $N$ - the number of samples selected. . . . .	135



## NOTATION

<b>AI</b>	Artificial Intelligence
<b>NLP</b>	Natural Language Processing
<b>NLG</b>	Natural Language Generation
<b>EARD</b>	Encode-Attend-Refine-Decode
<b>EAD</b>	Encode-Attend-Decode
<b>AQG</b>	Automatic Question Generation
<b>SD2T</b>	Structured Data to Text Generation
<b>QBAS</b>	Query Based Abstractive Summarization
<b>RNN</b>	Recurrent Neural Network
<b>LSTM</b>	Long Short Term Memory
<b>GRU</b>	Gated Recurrent Unit
<b>seq2seq</b>	Sequence to Sequence Models
<b>P2R</b>	PreliminaryDecoder-to-RefinementDecoder attention network
<b>E2R</b>	Encoder-to-RefinementDecoder attention network
<b>E2P</b>	Encoder-to-PreliminaryDecoder attention network
$\odot$	Hadmarad product
$\mathcal{L}$	Loss Function
$\eta$	Learning rate
$\theta$	Network parameters
$V$	Vocabulary Size
$P$	Sequence of words in the passage
$\mathbf{h}_i^p$	Passage level contextual representation for word at position $i$
$\mathbf{c}_t$	Passage-level context vector
$\mathbf{s}_t$	Decoder’s hidden state at timestep $t$
$\mathbf{e}_w$	Word Embedding function
$\mathbf{d}_t$	Refined context vector
$\mathcal{I}$	Wikipedia Infobox
$W$	Number of values in a Wikipedia Infobox
$\mathbf{c}_t^f$	Field-level context vector
$\kappa$	Cohen’s Kappa Score
$\mathbf{h}_t^d$	Refinement Decoder’s state at timestep $t$
$\tilde{\mathbf{h}}_t^d$	Preliminary Decoder’s state at timestep $t$
$\tilde{\mathbf{h}}_i^p$	Answer-aware representation for a passage word at position $i$
$\mathbf{h}^a$	Contextual representation of the answer
$\tilde{Q}$	Initial Draft of the question
$Q$	Refined Draft of the question





# CHAPTER 1

## Introduction

Natural Language Generation (NLG) deals with the problem of automatically generating human-understandable text from a given context. This context could be a natural language text itself, an image, a video, a structured table, *etc.* NLG includes a wide variety of tasks such as machine translation, abstractive summarization, dialogue generation, structured data to text generation, image captioning, automatic question generation, *etc.* It has always been one of the holy grails of Artificial Intelligence (AI) and is considered an essential trait for a machine to demonstrate Artificial General Intelligence. Indeed, the Turing Test, proposed way back in 1950, requires a machine to generate natural language text which is indistinguishable from that generated by humans. While significant progress has happened in NLG over the past 70 years, we are still nowhere close to building a machine that can comprehensively pass the Turing Test. Nevertheless, we have come a long way from the early rule based Machine Translation (MT) (Hutchins *et al.*, 1955) and summarization systems (Luhn, 1958) to the modern large scale transformer based models, which are trained on massive amounts of data (Radford *et al.*, 2018, 2019; Brown *et al.*, 2020).

Modern NLG systems are adept at a wide variety of tasks. For example, the GPT-3 model (Brown *et al.*, 2020) which is a massive transformer based model trained on 45 TB text data taken from multiple sources on the web, is capable of writing essays, poems, codes, jokes, summaries, *etc.* The evolution of NLG models into their modern day avatar of large transformer based models can be divided into four broad phases. The first phase includes the early rule based systems for translation (Hutchins *et al.*, 1955), summarization (Luhn, 1958) and dialogue generation (Weizenbaum, 1966). These systems relied on a few handcrafted rules and could generate responses for a limited set of inputs. The next important phase started with the arrival of statistical models (Brown *et al.*, 1990; Vogel *et al.*, 1996) which proposed a probabilistic model by casting NLG as the problem of generating a sequence  $t$  given an input  $c$ , which maximizes  $P(t|c)$ .

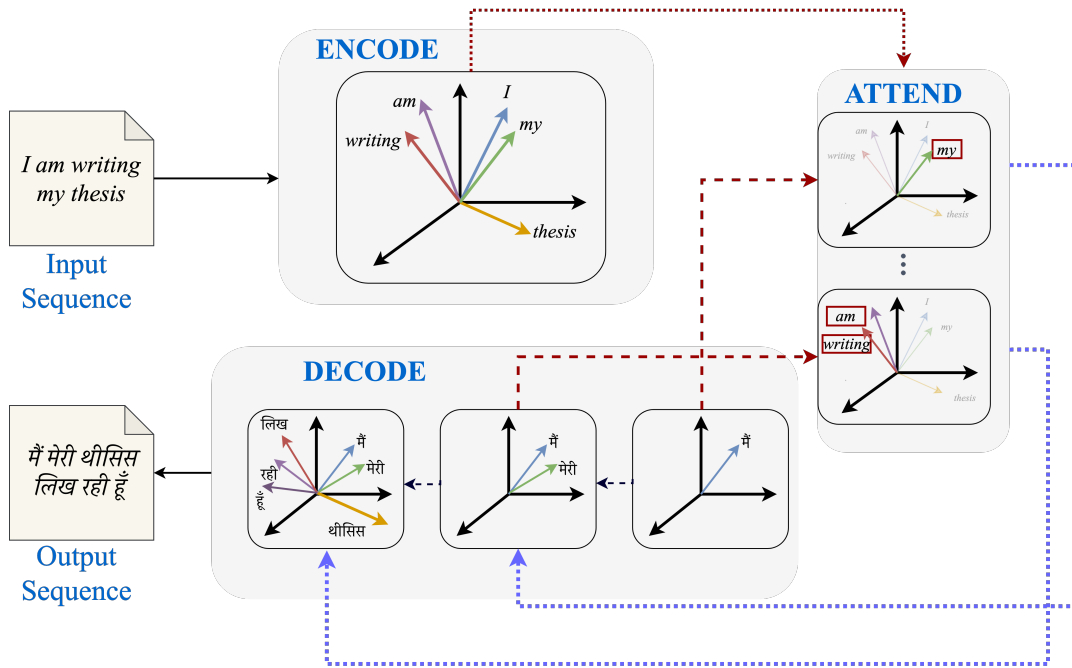


Figure 1.1: A basic seq2seq model consisting of encode-attend-decode modules. The encoder is responsible for learning meaningful contextual representations for words in the input. The decoder then predicts one word at a time by *attending to* relevant words in the input using the attend module.

These models dominated the field of NLG for a little over two decades till the arrival of Deep Learning. In particular, in this third phase, there was a transition from statistical models to end-to-end trainable neural network based models. An important innovation during this phase was the development of Recurrent Neural Network (RNN) based *encode-attend-decode* models (Bahdanau *et al.*, 2014). A key component of these models was an attention network which computes the importance of every token in the input sequence at each time step. While very successful, within a span of few years, RNNs were replaced by transformer based models, which are characterized by (i) multiple encoder layers, (ii) multiple decoder layers, and (iii) multiple attention networks.

The work done as a part of this thesis started during the third phase of the evolution of NLG when RNN based models with an attention network were the de facto standard for any NLG task (Rush *et al.*, 2015; Bahdanau *et al.*, 2014; Sankaran *et al.*, 2016; Du *et al.*, 2017; Mei *et al.*, 2016; Lebrete *et al.*, 2016; Nallapati *et al.*, 2016; Serban *et al.*, 2016b). These were also popularly known as *sequence-to-sequence* (seq2seq) models or *encode-attend-decode* models. An *encode-attend-decode* model as proposed in Bahdanau *et al.* (2014) contains (i) a RNN based encoder for computing contextual

representations of the tokens in the input sequence, (ii) a RNN based decoder that generates the output, one word at a time, and (iii) an attention network which computes how important an input token is for decoding the output at a particular timestep. The RNN could either be a vanilla RNN or a Gated Recurrent Unit (GRU) (Cho *et al.*, 2014) or a Long Short Term Memory (LSTM) unit (Hochreiter and Schmidhuber, 1997). Figure 1.1 shows the architecture of such an *encode-attend-decode* network for the task of Machine Translation. Despite their popularity and success, RNN based *encode-attend-decode* models have certain limitations. We focus on three such limitations in this thesis, as discussed in the next subsection.

## 1.1 Limitations of Sequence to Sequence Models

In Table 1.1 we show the output produced by a typical *encode-attend-decode* model for various NLG tasks and compare it with the expected gold standard output. Through these examples, we highlight three important limitations of such models which are addressed in this thesis.

First, from rows 1 to 4, we observe that the generated sequence often contains repeating phrases. This problem has been reported for a wide variety of tasks such as Abstractive Summarization (Suzuki and Nagata, 2017), Machine Translation (Sankaran *et al.*, 2016; Tu *et al.*, 2016), Open Ended Generation (Welleck *et al.*, 2019), Question Generation (Song *et al.*, 2017), *etc.* There could be several reasons for this, but in this work, we hypothesize that this could be due to the contextual representations passed to the decoder at every time step. In particular, at every time step, the decoder receives a contextual representation from the attention network. This contextual representation is a weighted sum of the representations of the tokens in the input sequence. If the decoder produces the same word at every timestep, could it be that it receives the same or similar contextual representations at those timesteps? We investigate this question in this thesis and propose solutions to alleviate this problem. While we do not formally show that the problem is due to similar representations being fed to the decoder at every time step, we do show that explicitly ensuring that the representations fed to the decoder at every time step are diverse, improves the quality of the generated text.

<b>Repeating Phrases</b>	
Abstractive Summarization (Suzuki and Nagata, 2017)	<p><b>Input:</b> China faces traditional powerhouse Germany on Tuesday in the round of ## at soccer 's world youth championship...</p> <p><b>Reference:</b> China success at youth world championship shows preparation for ##### Olympics.</p> <p><b>Generated:</b> China <b>germany germany germany germany</b> and <b>germany</b> at world youth championship</p>
Query-based Abstractive Summarization*	<p><b>Input:</b> <i>Passage:</i> Fuel cell critics point out that hydrogen is flammable, but so is gasoline... Also, gaseous hydrogen isn't the only method of storage under consideration–BMW is looking at liquid storage...</p> <p><i>Query:</i> are hydrogen fuel cell vehicles safe</p> <p><b>Reference:</b> hydrogen in cars is less dangerous than gasoline</p> <p><b>Generated:</b>hydrogen is <b>hydrogen hydrogen hydrogen</b> fuel energy</p>
Machine Translation (Sankaran <i>et al.</i> , 2016)	<p><b>Input:</b> die Teilnehmer der Proteste , die am Donnerstag um 6:30 AM morgens vor dem McDonald 's in der 40th Street und in der Madison Avenue begannen... .</p> <p><b>Reference:</b> Participants of the protest that began at 6.30 a.m. on Thursday near the McDonald's on 40th street and Madison Avenue...</p> <p><b>Generated:</b> The protests that began on Thursday at 06:30 before the <b>McDonald 's at McDonald 's at McDonald 's</b> on 40th Street and Madison Avenue.</p>
Open Ended Generation (Welleck <i>et al.</i> , 2019)	<p><b>Input:</b> ... Lyrically the song has excerpts of different languages including French , Spanish</p> <p><b>Reference:</b> , and German . In the first verse , the protagonist sings about being a “ girl who 's been in love with someone else ...</p> <p><b>Generated:</b> ,<b>Italian , Spanish , Italian , Spanish , Italian , Spanish , Spanish , Portuguese , Portuguese , Portuguese , Portuguese</b></p>
<b>Not Adequate</b>	
Structured Data to Text Generation*	<p><b>Input:</b> <i>NAME</i> &lt;Harrison B. Wilson Jr.&gt;, <i>PRECEDED_BY</i> &lt;Lyman Beecher Brooks&gt;, <i>BORN</i> &lt;April 21, 1925&gt;, <i>OCCUPATION</i> &lt;Educator, Professor, Coach&gt;</p> <p><b>Reference:</b> Harrison B. Wilson , Jr. (born april 21 , 1925) is an american educator , academic administrator.</p> <p><b>Generated:</b> <b>Lyman Wilson Jr.</b> (born april 21 , 1925) is an american educator and <b>educator.</b></p>
<b>Not Answerable</b>	
Question Generation*	<p><b>Input:</b> Liberated by Napoleon's army in 1806, Warsaw was made the capital of the newly created Duchy of Warsaw.</p> <p><b>Answer:</b> Napoleon</p> <p><b>Reference:</b> Whose army liberated Warsaw in 1806 ?</p> <p><b>Generated:</b> <b>What was the capital</b> of the newly duchy of Warsaw?</p>

Table 1.1: Samples of erroneous output produced by encode-attend-decode models for various NLG tasks. The generated outputs have been taken from the respective mentioned works. \* refers to the encode-attend-decode models taken as baselines in this thesis.

The above problem was a **task-agnostic problem**, in the sense that it can occur for any NLG task. We now look at two **task-specific problems** with RNN based *encode-attend-decode* models. We first consider the task of generating natural language descriptions from structured data. Here, the input is a table of facts arranged in rows. Each row corresponds to a field, such as name, occupation, birthdate, *etc.* Further, each field can have one or more fields associated with it (*e.g.*, the occupation field can have multiple values such as *professor, coach, etc.*). Row 5 of Table 1.1 shows an example of this task and highlights a potential problem with *encode-attend-decode* models. First, in the generated output the decoder seems to jump from one field to another instead of staying on just one field; *e.g.*, it mixes *Lyman* and *Wilson* which belong to two different fields. Second, it does not cover all values from a given field; *e.g.*, it does not list down all the occupations. The vanilla *encode-attend-decode* models are task-agnostic and clearly do not have niche components that can capture such task specific behaviour. We propose a refinement to the vanilla *encode-attend-decode* by adding special components which capture this task specific behaviour. In particular, our components ensure that the decoder stays on a field as long as required (*i.e.*, till all the data from that field is covered) and never looks back at a field once it is covered (thereby ensuring that data from two different fields is not mixed).

Lastly, we consider another **task-specific problem** in the context of Automatic Question Generation (AQG). Here, given an input passage and a specific answer, the task is to generate a question that can not only be answered from the given passage but also leads to the specific answer provided as input. Thus, unlike other NLG tasks such as machine translation, dialogue generation and abstractive summarization, where the generated text is required to be fluent, coherent, factual, *etc.*, in AQG there is an additional requirement that the generated question should be *answerable*. For example, consider the generated question shown in row 6 of Table 1.1. It is neither answerable from the given passage, nor does it lead to the specific answer provided as input. To capture this task specific characteristic we first need to propose a metric which captures **answerability**, *i.e.*, given a reference question and a generated question it assigns a score on a scale of 0 to 1, indicating how close is the the generated question in terms of answerability to the reference question. Next, we need to ensure that the model is rewarded for generating questions which score high on this metric. Vanilla *encode-*

*attend-decode* models which are trained to maximise log likelihood, clearly do not use such task specific criteria such as *answerability* as objective functions. In this work, we address this issue by proposing a two stage decoder wherein the second stage refines the output of the first stage to maximise *answerability* of the generated question.

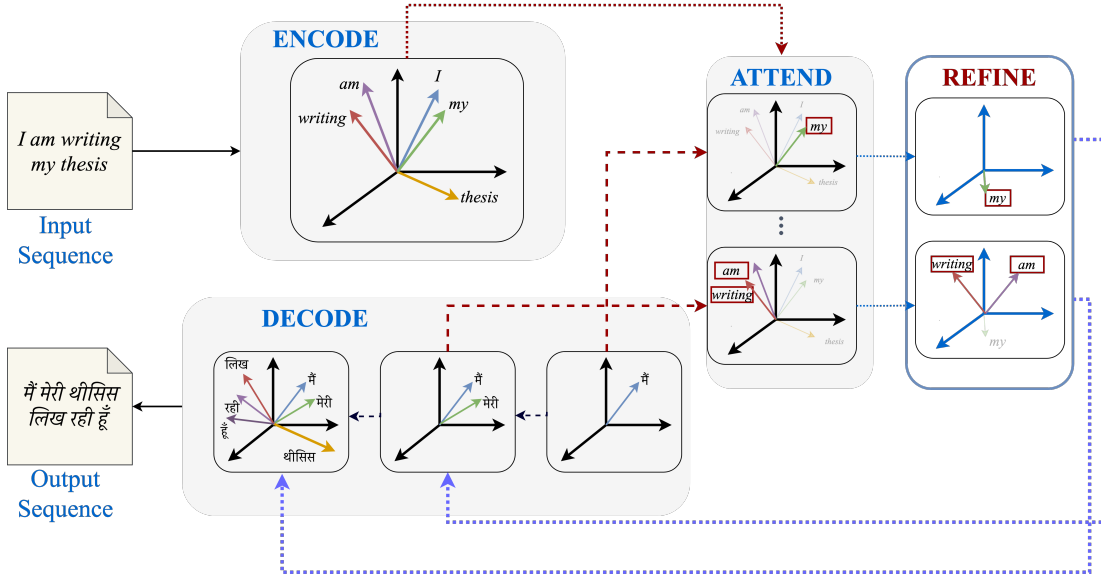


Figure 1.2: Seq2seq architecture with proposed refine module. The refine module enriches the attended input representations computed from attend module and then passes it to the decode module.

## 1.2 Objectives of the Thesis

This thesis aims at improving the performance of *seq2seq* models by alleviating the task-agnostic and task-specific problems listed above and summarised below.

- *Mitigating the task agnostic problem of repeating phrases in the output:* Our first objective is to mitigate the repeating phrase problem for query-based abstractive summarization by diversifying the contextual representations.
- *Modeling task-specific characteristics for structured data to text generation:* Our second objective is to model task-specific characteristics for structure data to text generation, to enhance the fluency and adequacy of the generated descriptions.
- *Modeling task-specific characteristics for question generation:* Our third objec-

tive is to increase the answerability of generated questions based on a given passage and a specific answer.

### 1.3 Contributions of this Thesis

A common theme in this thesis is that the vanilla encode-attend-decode models need some refinement to address task specific and task agnostic problems. To do so, we introduce a **refine** module, the specifics of which depend on the task and problem being addressed. For example, for solving the problem of repeating phrases, the *refine* module contains a component which orthogonalizes and thus diversifies the context representations passed to the decoder at each time step. On the other hand, to generate answerable questions, the refine module contains an additional decoder which makes the outputs of the first decoder more answerable. This gives rise to a generic framework which we refer to as the **encode-attend-refine-decode** which could potentially be extended to handle other task-specific and task-agnostic limitations of vanilla encode-attend-decode models. In the context of this framework, we make the following contributions in this thesis.

- As our first contribution, we address the problem of *repeating phrases* in the output of NLG systems. To this end, we propose a refine module to diversify the context vectors generated at each time step. In particular, we propose a Diversity LSTM cell, a novel variant of the basic LSTM cell. It ensures that the LSTM network’s output is orthogonal to the cell-state, which in turn ensures that the context vector at each time step is distinct, thereby reducing repetitions in the generated text. Further, to address situations where repetitions are naturally required (*e.g.*, for generating an output of the form “*He kept talking and talking and talking*”, we introduce a soft orthogonalization parameter which allows the network to choose *if* and *how much* to diversify the context vectors. To evaluate this model, we introduce a new query-based abstractive summarization dataset based on Debatepedia. Our experiments show that with the refine module, the proposed model outperforms vanilla encode-attend-decode models with a gain of 28%(absolute) in ROUGE-L scores.

- As our second contribution, we address the problem of generating natural language descriptions from a structured table of facts containing fields (such as nationality, occupation, *etc.*) and values (such as Indian, actor, director, *etc.*). Previous works have adapted vanilla seq2seq models for this task, by hierarchically encoding the table. In particular, they first computed a representation for each field by combining the representations of the values in that field, and then computed a representation for the table by combining the representations of the fields. An attention network then pays attention to the relevant fields and feeds a weighted contextual representation to the decoder at each time step. However, such models do not mimic the behaviour of humans while generating the captions. For example, a human would *stay on* a field until the relevant information from that field has been written. Moreover, she will *never look back* to that field while generating the rest of the description. To simulate these two behaviors: *stay-on* and *never-look-back*, we design the refine module with a gated orthogonalization mechanism. It comprises of a soft learnable gate which decides at each timestep of the decoder whether to *stay on* the same field or *never-look-back* at a field. If the latter behavior is preferred, then the new context vector is orthogonalized to the current one to ensure that the same field is not attended to again. We experiment with a dataset released by Lebret *et al.* (2016) which contains fact tables about people and their corresponding one-line biographical descriptions in English and show that our model gives 2.2% relative improvement over the encode-attend-decode paradigm based baseline. In addition, we also validate our proposed model on similar datasets as introduced in Lebret *et al.* (2016) for French and German Languages. We observe a relative improvement of 12% and 14% over the encode-attend-decode paradigm based baseline.
- As our third contribution, we address the problem of improving the *answerability* of questions generated from a given passage. A key issue here is that existing metrics for evaluating AQG systems are adopted from other NLG tasks such as Machine Translation and hence are not adequate for evaluating AQG-specific characteristics. Hence, there is no reward for a model to generate answerable questions. As a first step, we show that existing metrics used for evaluating AQG systems do not focus on answerability of the generated question. In particular,



they do not explicitly prefer questions which contain all relevant information such as question type (Wh-types), entities, relations, etc. and thus do not always correlate well with human judgments about answerability of a question. To alleviate this problem, we introduce a scoring function to capture answerability and show that when this scoring function is integrated with existing metrics, they correlate significantly better with human judgments. Having introduced this metric, we observe that existing AQG systems often do not produce answerable questions. In particular, the generated questions look like an incomplete draft of the desired question with a clear scope for refinement. To alleviate this shortcoming, we propose a method which tries to mimic the human process of generating questions by first creating an initial draft and then refining it. More specifically, we propose an encode-attend-refine-decode paradigm-based model which contains two decoders, *viz.*, Preliminary and Refinement Decoders. The Refinement Decoder pays attention to both (i) the original passage and (ii) the question (initial draft) generated by the Preliminary Decoder. In effect, it refines the question generated by the Preliminary Decoder, thereby making it more correct and complete. We evaluate our proposed model on multiple datasets and show that it outperforms existing state-of-the-art methods by 7-16% on all of these datasets. Lastly, we show that we can improve the quality of the Refinement Decoder on specific metrics, such as fluency and answerability, by explicitly rewarding revisions that improve on the corresponding metric during training.

## 1.4 Outline of the Thesis

This chapter gives a brief overview of the problems addressed and the solutions proposed in this thesis. The rest of the thesis is organized as follows:

- In Chapter 2, we provide the background by introducing all the basic concepts required to understand the work presented in this thesis. In particular, we give an overview of neural networks, recurrent neural networks and some of its variants and finally, vanilla sequence to sequence models.
- In Chapter 3, we first describe how vanilla seq2seq models have been adopted

for different tasks addressed in this work. We then discuss parallel works which address the same limitations as discussed in this thesis. Finally, we discuss related works that use similar concepts as used in our thesis to mitigate other problems encountered in NLG.

- In Chapter 4, we present our encode-attend-refine-decode architecture for mitigating the problem of repeating phrases in the output. We then present experimental results comparing our model to vanilla seq2seq models on a new dataset for query-based abstractive summarization.
- In Chapter 5, we present an encode-attend-refine-decode architecture for the task of structured data to text generation. We discuss our method which mimics the *stay on* and *never look back* behaviour. We also present experimental results comparing our model to state of the art models for this task on a dataset comprising of biographies taken from Wikipedia.
- In Chapter 6, we first propose a metric designed for evaluating question generation systems with a focus on *answerability*. We then propose a model based on the encode-attend-refine-decode paradigm, which uses answerability as an explicit reward signal to refine the outputs of the Preliminary Decoder. We then present experimental results comparing our model to state of the art models for this task on a wide variety of tasks.
- In Chapter 7, we present important conclusions from our work, and highlight some interesting future research directions.

# CHAPTER 2

## Background

In this chapter, we introduce the basic concepts required to understand the work done as a part of this thesis. We first briefly explain the different NLG tasks that we considered in this work. We then give an overview of Neural Networks, LSTMs, Sequence-to-Sequence Models, and Attention Mechanisms. Finally, we discuss the evaluation metrics used in this work.

### 2.1 Relevant NLG Tasks

In this work, we consider three NLG tasks, *viz.*, abstractive summarization, structured data to text generation and automatic question generation. We describe each of these tasks below.

#### 2.1.1 Abstractive Summarization

Given a document, the task here is to generate a human-like summary. We show a sample document and possible summaries in Table 2.1. It is obvious that the summaries are expected to be concise, coherent, grammatically correct, and faithful to the information present in the given document. Further, just like any other NLG task, multiple correct answers (summaries) exist for a given document.

#### 2.1.2 Query-Based Abstractive Summarization

Given a document and a query, the task is to generate a summary focused on addressing the given query. Just like abstractive summarization, we expect the summary to be concise, coherent, and faithful to both document and the query in this task. We show an example of this task in Table 2.2.

---

---

**Document:** Babar Azam made a total of 228 runs in Pakistan’s famous series win against South Africa, including a century in the first match in Centurion. That helped him move ahead of Kohli - who has been No.1 since displacing AB de Villiers in October 2017 - by a point, but he dropped to 852 after his score of 31 in the second ODI and remained at No. 2 by the time of the last weekly rankings update. After his knock in the third ODI, he gained eight more than Kohli, to become just the fourth Pakistan batsman to attain the top ranking after Zaheer Abbas (1983-84), Javed Miandad (1988-89), and Mohammad Yousuf (2003).

---

**Summaries:**

1. Babar Azam becomes only fourth Pakistan player to attain top ranking as ODI batsman, after Zaheer Abbas, Javed Miandad and Mohammad Yousuf .
  2. Babar Azam displaced Kohli, who had earlier displaced AB de Villiers in 2017, to become the No.1 ranked ODI batsman.
  3. Pakistan wins series against South Africa .
- 

Table 2.1: Example of a document and different possible manually generated summaries for the given document.

---

---

**Document:** Babar Azam made a total of 228 runs in Pakistan’s famous series win against South Africa, including a century in the first match in Centurion. That helped him move ahead of Kohli - who has been No. 1 since displacing AB de Villiers in October 2017 - by a point, but he dropped to 852 after his score of 31 in the second ODI and remained at No. 2 by the time of the last weekly rankings update. After his knock in the third ODI, he gained eight more than Kohli, to become just the fourth Pakistan batsman to attain the top ranking after Zaheer Abbas (1983-84), Javed Miandad (1988-89), and Mohammad Yousuf (2003).

**Query:**

**Summary:**

Babar Azam becomes only fourth Pakistan player to attain top ranking as ODI batsman, after Zaheer Abbas, Javed Miandad and Mohammad Yousuf .

---

Table 2.2: Example of a summary for document in Table 2.1 which is specific to the given query.

---



---

<b>Charles B. Winstead</b>	
<b>Born</b>	May 25, 1891 Sherman, Texas, US
<b>Died</b>	August 3, 1973 (aged 82) Albuquerque, New Mexico, US
<b>Occupation</b>	FBI agent
<b>Employer</b>	FBI
<b>Known for</b>	Shooting John Dillinger
<b>Title</b>	Special agent

---

### Descriptions

1. Charles B. Winstead (May 25, 1891 – August 3, 1973) was an FBI agent in the 1930s–40s.
  2. Charles B. Winstead was famous for being one of the agents who shot and killed John Dillinger
- 

Table 2.3: Example of a Wikipedia Infobox (structured data) and different possible descriptions of the Infobox.

## 2.1.3 Structured Data to Text Generation

In both the tasks that we saw earlier, the input was a natural language text (or texts). We now consider a task where the desired output is a natural language text, but the input is structured data, *i.e.*, it is not a natural language text. More specifically, the input is organized in a well-defined table with rows and columns (similar to a typical database table or an Infobox in a Wikipedia page). The task then is to generate a natural description from this table of facts. Apart from being grammatically correct, the generated description must be factually correct and faithful to the information in the structured data. Table 2.3 shows a sample Infobox (containing structured data) and potential descriptions of the Infobox.

## 2.1.4 Question Generation

Given a document and optionally a specific answer, the task here is to generate a question that can be answered from the given document and leads to the given answer. The generated question should be unambiguous, grammatically correct, and answerable from the given document. Table 2.4 shows examples of different questions that can be generated from the given document when no specific answer is given as input.

---

**Passage:** Oxygen is used in cellular respiration and released by photosynthesis, which uses the energy of sunlight to produce oxygen from water.

---

**Questions**

1. What life process produces oxygen in the presence of light?
  2. Photosynthesis uses which energy to form oxygen from water?
  3. From what does photosynthesis get oxygen?
- 

Table 2.4: Example of a document and different questions that can be generated from it. These examples are taken from SQuAD dataset (Rajpurkar *et al.*, 2016).

---

**Passage:** Oxygen is used in cellular respiration and released by photosynthesis, which uses the energy of sunlight to produce oxygen from water.

**Answer:** *sunlight*

**Question:** Photosynthesis uses which energy to form oxygen from water?

---

Table 2.5: Example of a question corresponding to the given (document, answer) pair from SQuAD dataset (Rajpurkar *et al.*, 2016).

On the other hand, Table 2.5 shows a specific question that can be generated when the desired answer is also provided as input.

## 2.2 An overview of Neural Networks

### 2.2.1 Artificial Neuron

Artificial neural networks (ANN) are motivated from biological neural networks, which are a densely connected mesh of biological neurons. The most fundamental unit of an

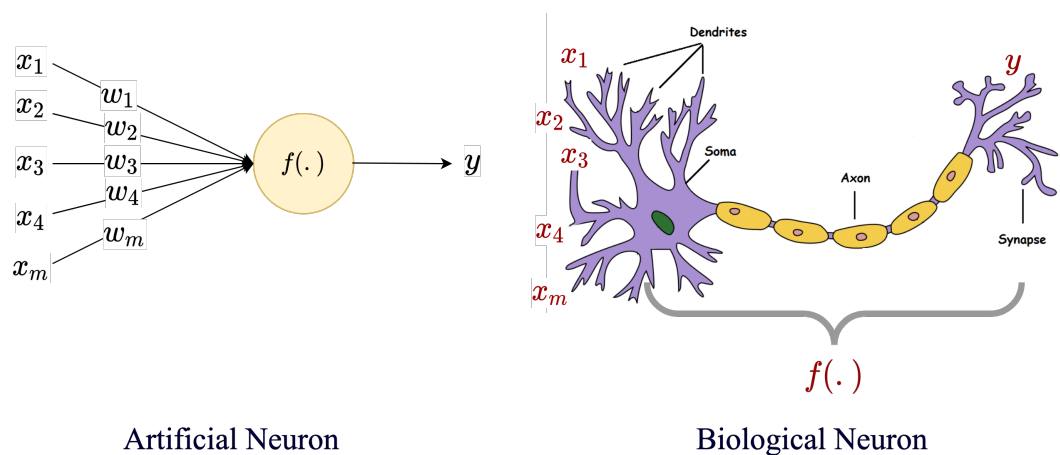


Figure 2.1: An artificial neuron (left) and a biological neuron (right).

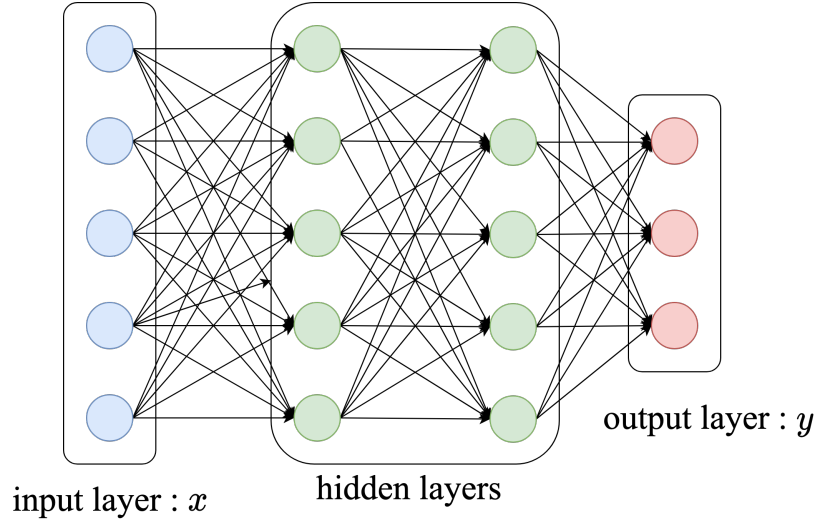


Figure 2.2: A simple feed-forward neural network with two hidden layers.

ANN is an artificial neuron which is a mathematical model inspired by a biological neuron as illustrated in Figure 2.1. Given a set of inputs  $\mathbf{x} = \{x_1, \dots, x_m\}$  the output of a neuron is given by  $g(\mathbf{x}) = \sum w_i x_i$ . The weights  $\{w_1, \dots, w_m\}$  are used to weigh the different inputs connected to the neuron. Note that  $g$  can be any function and is typically chosen to be a non-linear function. We will further see that the weights can be learned such that the model produces the desired output.

### 2.2.2 Artificial Neural Networks

An ANN is a network of artificial neurons as shown in Figure 2.2. Typically, there are multiple layers in the network, with each layer receiving input from the previous layer and feeding its output to the next layer. Note that the initial and final layers are referred to as input and output layers respectively, whereas the intermediate layers are referred to as hidden layers. The output of the  $i$ -th layer is denoted by  $\mathbf{h}_i$  and is given by

$$\mathbf{a}_i = \mathbf{W}_i \cdot \mathbf{h}_{i-1} + \mathbf{b}_i \tag{2.1}$$

$$\mathbf{h}_i = \sigma(\mathbf{a}_i) \tag{2.2}$$

where  $\mathbf{W}_i$ ,  $\mathbf{b}_i$  denote the weight and bias in the  $i$ -th layer. Note that  $\mathbf{h}_0 = \mathbf{x}$ , where  $\mathbf{x}$  is the input.  $\sigma$  is the activation function and is typically a non-linear function such as the logistic function, tanh function, ReLU function (Agarap, 2018), *etc.*

### 2.2.3 Training ANNs

Given a training set  $\{\mathbf{x}_i, y_i\}_{i=1}^N$ , the goal of a neural network with parameters  $\theta$ , is to learn a function  $f_\theta(\mathbf{x})$  such that  $f_\theta(\mathbf{x}_i)$  is as close to  $y_i$  as possible. To do so, we define a loss function,  $\mathcal{L}(\theta)$ , which captures the difference between  $f_\theta(\mathbf{x})$  and  $y$ . Two popular loss functions that are commonly used are the mean square error for regression and the cross entropy loss for classification as defined below.

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i, \theta))^2 \quad \text{Mean Square Error} \quad (2.3)$$

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^N y_i \log(f(\mathbf{x}_i, \theta)) \quad \text{Cross-Entropy Loss} \quad (2.4)$$

Having defined a loss function, the goal then is to find the parameters  $\theta$  which minimize this loss function. This is thus an optimization problem, and a popular way of finding  $\theta$  is to use the gradient descent algorithm with back-propagation. This is an iterative algorithm where starting from an initial random value for  $\theta$ , at each stage a new value is computed using the following update rule:

$$\theta = \theta - \eta \nabla_\theta L(\theta) \quad (2.5)$$

where  $\eta$  is the learning rate and  $\nabla_\theta$  is the gradient of the loss function *w.r.t.*  $\theta$ .

For gradients computation Rumelhart *et al.* (1986) proposed the back-propagation algorithm. It allows the backward flow of information from the cost function to the network parameters which need to be updated. Figure 2.3 illustrates one such example of backward flow to the parameter  $w$  in the initial layer of the network. This algorithm uses the chain rule to compute the gradients. Note that the gradient descent algorithm is sensitive to the choice of the learning rate  $\eta$ . A lower learning rate can lead to slower convergence, or a higher learning rate can lead to oscillations. Several variants of gradient descent (GD) such as Momentum based GD, Nesterov Accelerated GD (Nesterov, 1983), RMSProp (Hinton, 2016), AdaGrad (Duchi *et al.*, 2011), Adam (Kingma and Ba, 2014), *etc.*, are used to overcome the limitations of vanilla gradient descent. Some



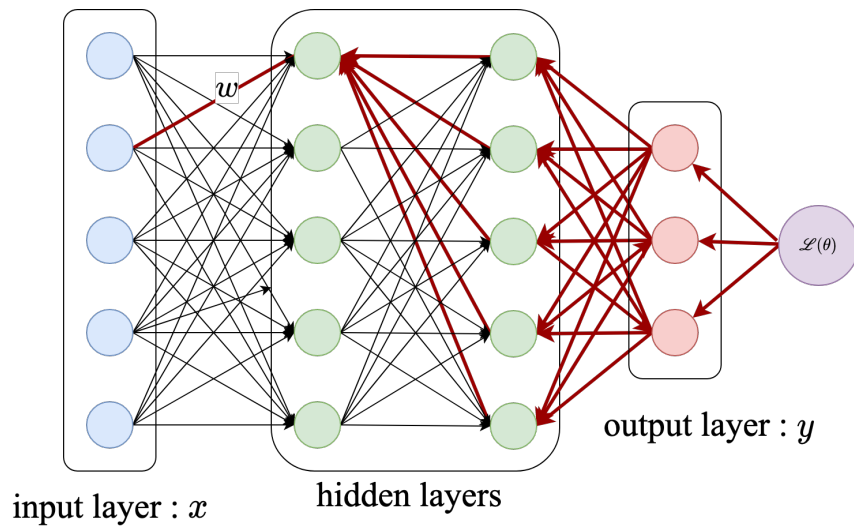


Figure 2.3: Backward flow of error for weight  $w$  for a sample feedforward network with two hidden layers.

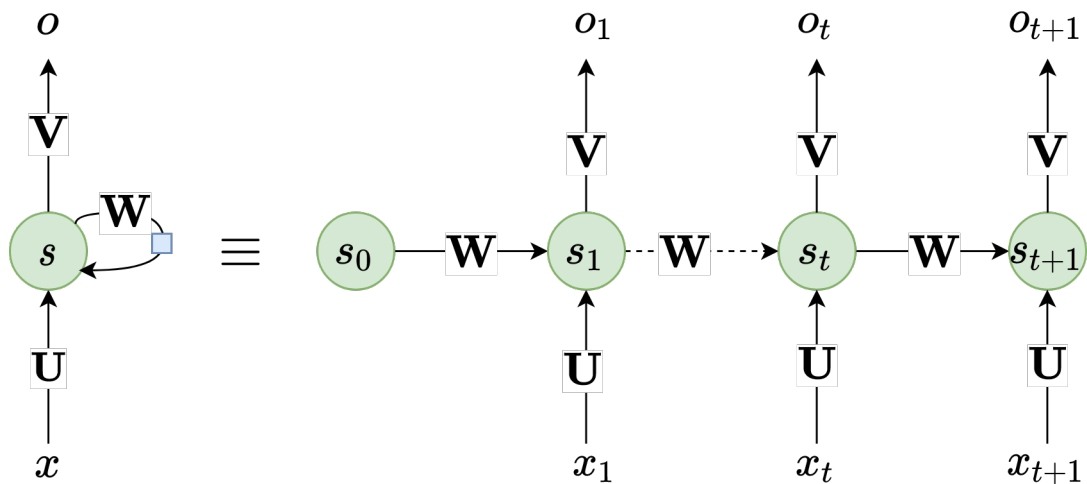


Figure 2.4: Recurrent Neural Network

of these algorithms use an adaptive learning rate which leads to faster convergence and improved performance.

## 2.3 Recurrent Neural Networks

In many real world applications, the input is in the form of a sequence. For example, a text is a sequence of words, a video is a sequence of frames, and so on. More formally, the input can be divided into  $T$  timesteps and the output at timestep  $t$  may depend on all inputs up to that point. Such dependence between elements appearing in a sequence can be modeled using a Recurrent Neural Network as shown in Figure 2.4. For a given

input sequence  $\mathbf{x} = \{x_1, \dots, x_N\}$ , and model parameters  $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{b}_v, \mathbf{b}_w$  the output at every timestep  $k$  is computed as follows:

$$\begin{aligned}\mathbf{o}_k &= f(\mathbf{V} \cdot \mathbf{s}_k + \mathbf{b}_v) \\ \mathbf{s}_k &= g(\mathbf{W} \cdot \mathbf{s}_{k-1} + \mathbf{U} \cdot \mathbf{e}(x_k) + \mathbf{b}_w)\end{aligned}\tag{2.6}$$

Here  $\mathbf{e}(x_k)$  is  $d$ -dimensional representation for token  $x_k$ . The output function  $f(\cdot)$  depends on the task at hand. For example, for a classification task, it could be the softmax function. The function  $g$  is a non-linear function such as the logistic function or the tanh function. The intermediate layer often referred to as *state*,  $\mathbf{s}$ , is the associative memory representation that is updated through the recurrent connection. Therefore, the state at  $k^{th}$  timestep is a function of state at the  $k - 1^{th}$  timestep and the input at the  $k^{th}$  timestep. The output could either be generated at each timestep or only at the final timestep, depending on the task. For example, if the task is to process a text and predict the sentiment then the output,  $\mathbf{o}_T$ , only needs to be computed at the final timestep  $T$ . On the other hand, if the task is to predict the part of speech tag of every word in the sequence then the output,  $\mathbf{o}_t$ , will be computed at every timestep  $t$ .

### 2.3.1 Learning Algorithms for RNNs

**Backward Propagation of error Through Time (BPTT)** (Werbos, 1990) is the most popular algorithm for training RNNs. It can be thought of as a generalization of the back-propagation algorithm used for training feedforward neural networks and works by temporally unrolling the network. For a given weight  $\mathbf{W}$ , the gradient needed for the update rule can be computed using the chain rule as shown below.

$$\frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{o}_N} \frac{\partial \mathbf{o}_N}{\partial \mathbf{s}_N} \frac{\partial \mathbf{s}_N}{\partial \mathbf{W}}\tag{2.7}$$

$$\frac{\partial \mathbf{s}_k}{\partial \mathbf{W}} = \frac{\partial \mathbf{s}_k}{\partial \mathbf{s}_{k-1}} \frac{\partial \mathbf{s}_{k-1}}{\partial \mathbf{W}}\tag{2.8}$$

$$\frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{o}_N} \frac{\partial \mathbf{o}_N}{\partial \mathbf{s}_N} \frac{\partial \mathbf{s}_N}{\partial \mathbf{s}_{N-1}} \cdots \frac{\partial \mathbf{s}_k}{\partial \mathbf{s}_{k-1}} \cdots \frac{\partial \mathbf{s}_1}{\partial \mathbf{W}} \quad (2.9)$$

$$= \frac{\partial \mathcal{L}(\theta)}{\partial \mathbf{o}_N} \frac{\partial \mathbf{o}_N}{\partial \mathbf{s}_N} \prod_{i=2}^N \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} \frac{\partial \mathbf{s}_1}{\partial \mathbf{W}} \quad (2.10)$$

Due to the product term,  $\prod_{i=2}^N \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} \frac{\partial \mathbf{s}_1}{\partial \mathbf{W}}$ , the gradients can either vanish or explode depending on the spectral radius of the recurrent weight  $\mathbf{W}$ . This is commonly referred to as the problem of vanishing or exploding gradients. Needless to say, this severely affects the training of RNNs, especially as the length of the input sequence and thus the number of terms in the product increases. This limitation was first discussed in Bengio *et al.* (1994) and then was further studied in Pascanu *et al.* (2012).

## 2.4 Long Short Term Memory

To overcome the problem of vanishing and exploding gradient, in 1997, Hochreiter and Schmidhuber proposed Long Short Term Memory. The key idea was to have a linear relationship between the recurrent states  $(\mathbf{s}_i, \mathbf{s}_{i-1})$ , which then allows a constant error flow in the network, preventing the gradients from exploding or vanishing. The gradient computation will not vanish and let the gradient flow unchanged. The gradient flow inside and outside this cell state was restricted using learnable gates, *viz.*, input and output gates. These gates essentially provide read and write access to the associative memory modeled in the LSTM cell. The LSTM cell was further modified to include a learnable forget gate (Gers *et al.*, 1999). The forget gate helps in resetting the LSTM state whenever required. The architecture shown in Figure 2.5 is a vanilla LSTM model and is widely used in various NLP tasks.

For a given input sequence  $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ , and timestep  $t$ , the gates and

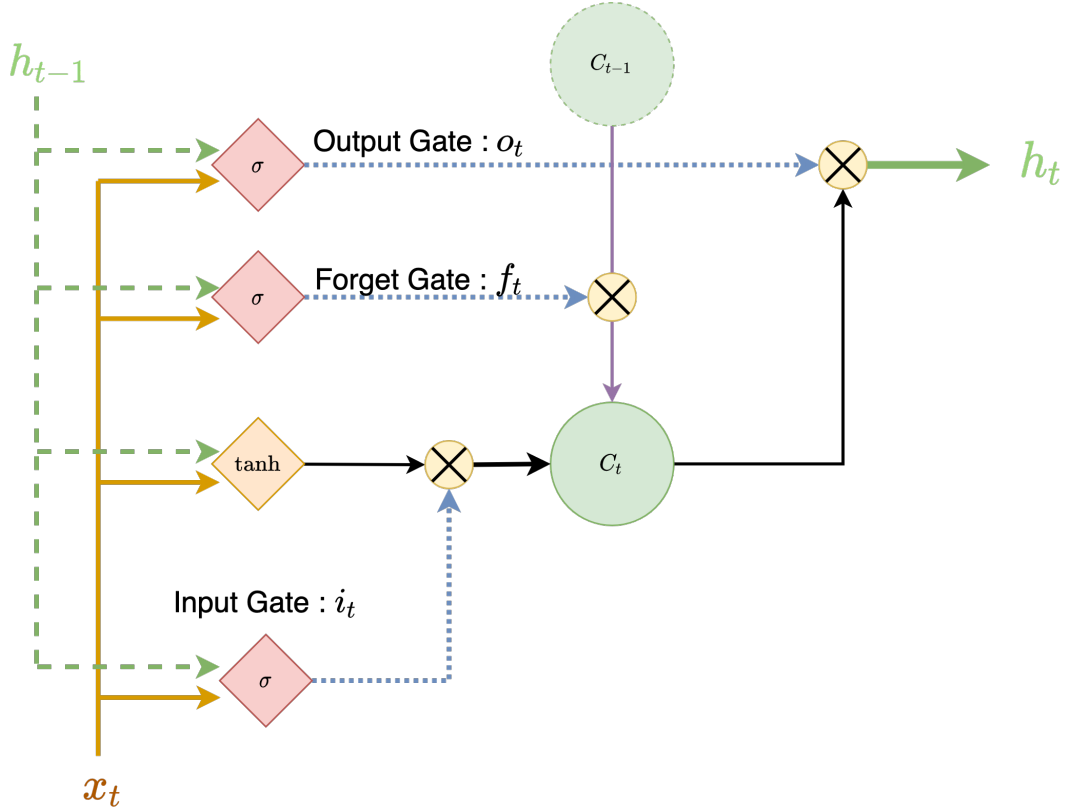


Figure 2.5: The computations in one timestep of a Long Short Term Memory Cell.

states in an LSTM are computed using the following equations:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot \mathbf{e}(x_t) + \mathbf{U}_i \cdot \mathbf{h}_{t-1} + \mathbf{b}_i) \quad (2.11)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot \mathbf{e}(x_t) + \mathbf{U}_f \cdot \mathbf{h}_{t-1} + \mathbf{b}_f) \quad (2.12)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot \mathbf{e}(x_t) + \mathbf{U}_o \cdot \mathbf{h}_{t-1} + \mathbf{b}_o) \quad (2.13)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \cdot \mathbf{e}(x_t) + \mathbf{U}_c \cdot \mathbf{h}_{t-1} + \mathbf{b}_c) \quad (2.14)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad (2.15)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (2.16)$$

Here  $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$  are the input, forget and output gates respectively. These gates provide read, delete and write access to the network's cell state. Here  $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o \in \mathbb{R}^{m \times d}$ ,  $\mathbf{U}_i, \mathbf{U}_f, \mathbf{U}_o, \mathbf{U}_c \in \mathbb{R}^{m \times m}$ , and  $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_c \in \mathbb{R}^m$  are network parameters. Note that  $\mathbf{e}(x_t)$  represents a  $d$ -dimensional representation for token  $x_t$ , and  $m$  is the cell-state size, often referred to as the hidden size of the LSTM cell.

*Gated Recurrent Units (GRUs):* GRUs combine the forget and input gate into one

update gate and remove the distinction between hidden and output states. The gates and states of a GRU are computed as follows:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \cdot \mathbf{e}(x_t) + \mathbf{U}_z \cdot \mathbf{h}_{t-1} + \mathbf{b}_z) \quad (2.17)$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \cdot \mathbf{e}(x_t) + \mathbf{U}_r \cdot \mathbf{h}_{t-1} + \mathbf{b}_r) \quad (2.18)$$

$$\tilde{\mathbf{h}} = \tanh(\mathbf{W}_c \cdot \mathbf{e}(x_t) + \mathbf{r}_t \odot \mathbf{U}_c \cdot \mathbf{h}_{t-1} + \mathbf{b}_c) \quad (2.19)$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}} \quad (2.20)$$

Note that in this thesis, we use either LSTM or GRU to learn meaningful representations for the input sequence and/or generate the output sequence. Additionally, some of our primary contributions are in modifying the basic LSTM cell, as discussed above. Throughout the rest of the chapter, when we use the term RNN, it could be replaced by LSTM and GRU also.

## 2.5 Sequence-to-Sequence Models

In general, a sequence-to-sequence model could refer to any model which takes a sequence as input and generates a sequence as an output. In particular, this term was popularly used to describe RNN based models, which use an RNN to encode the input sequence and then another RNN to decode the output, one word at a time. The model takes as input, a sequence  $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$  of length  $N$  and generates a target sequence  $\mathbf{y} = \{y_1, y_2, \dots, y_M\}$  of length  $M$ . It contains two modules, *viz.*, encoder and decoder as described below.

### 2.5.1 Encoder Module

The encoder module in an RNN based seq2seq model typically consists of two layers, an embedding layer and a contextual layer.

#### Embedding Layer

*Word Level Embedding:* In this layer, every word in the input sequence is converted to a  $d$ -dimensional representation or embedding. Typically, the embedding is obtained

from a pretrained word embedding model, such as Glove (Pennington *et al.*, 2014). Throughout this thesis, we refer to this embedding function as  $e(\cdot)$ , which generates a  $d$ -dimensional representation for a given word,  $w_i$ .

*Character Level Embedding:* Zhang *et al.* (2015) proposed a mechanism to learn word embeddings using the embeddings of the characters that compose the word. A character-level embedding is an effective strategy to learn meaningful representations for rare words in the training corpus or words that are out of the vocabulary. Similar to Seo *et al.* (2016), in this thesis, we use Convolutional Neural Network to learn character-level embeddings. We use 1D convolution filters with a receptive field of  $\{2, 3, 5\}$  to capture character  $n$ -gram information from a word.

**Contextual Layer** In this layer, we encode the interaction of each word with its surrounding words or context. We learn such contextual representation for every word in the input sequence using a bidirectional LSTM.

$$\vec{\mathbf{h}}_t = \text{LSTM}(e(w_t), \vec{\mathbf{h}}_{t-1}) \quad \forall t \in [1, N] \quad (2.21)$$

where  $\vec{\mathbf{h}}_t$  is the hidden state of the forward LSTM at time  $t$ , *i.e.*, when the sequence is parsed from left-to-right. Similarly, we compute hidden states  $\overleftarrow{\mathbf{h}}_t$  from a backward LSTM which processes the sequence from right-to-left. We then concatenate the forward and backward hidden states as  $\mathbf{h}_t = [\vec{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$ , that gives the contextual representation of the word,  $w_t$ . We represent the set of contextual representations for all words in the input sequence as  $\mathbf{H} = \{\mathbf{h}_1, \dots, \mathbf{h}_N\}$ . The representation of the overall sequence  $\mathbf{x}$  is typically taken as the contextual representation of either the last or the first token of the sequence, *i.e.*, either  $\mathbf{h}_1$  or  $\mathbf{h}_N$ . For ease of notation, we represent the contextual representation of the overall context as  $\mathbf{c}$ .

## 2.5.2 Decoder Module

The decoder module generates the target output sequence, one word at a time. It is essentially a language modeling task, where the current word depends on the previously generated words as well as the input context. The model is trained to maximize the

probability of generating the target sequence,  $\mathbf{y}$ :

$$\mathbf{y}^* = \arg \max_{\mathbf{y}} \prod_{t=1}^m \tilde{p}(\tilde{y}_t | \tilde{y}_{t-1}, \dots, \tilde{y}_1, \mathbf{c}) \quad (2.22)$$

Here,  $\tilde{p}$  models the conditional probability distribution of a word at time  $t$  given the previous words and the context. We model this distribution using an LSTM as shown below.

$$\tilde{\mathbf{s}}_t, \mathbf{s}_t = \text{LSTM}(\mathbf{s}_{t-1}, [\mathbf{e}(y_{t-1}) : \mathbf{c}]) \quad (2.23)$$

$$y_t = \text{softmax}(\mathbf{W}_o \cdot \tilde{\mathbf{s}}_t) \quad (2.24)$$

Here,  $\mathbf{c}$  is the contextual representation obtained from the encoder, as explained above.  $\tilde{\mathbf{s}}_t$  represents the output from the LSTM cell, and  $\mathbf{s}_t$  is the internal cell state of the LSTM.  $\mathbf{W}_o \in \mathbf{R}^{|V| \times l}$  is used to linearly transform the output state to a vector whose size is the same as the number of words in the vocabulary. This vector is then converted into a probability distribution,  $y_t$ , using the softmax function.  $[:]$  represents the concatenation operation, and  $\mathbf{e}(y_{t-1})$  represents the embedding of the word which has the highest probability under the distribution  $y_{t-1}$ .

While seq2seq models, as described above, were successful for many NLG tasks, several studies (Cho *et al.*, 2014; Pascanu *et al.*, 2012) showed that they are not adequate for longer input sequences. In particular, the fixed-size representation in a RNN or LSTM state is unable to capture all the relevant information in the longer input sequence. To overcome this problem, the attention mechanism was proposed, as described below.

## 2.6 Attention Mechanisms

The attention mechanism was originally proposed in the context of Machine Translation by Bahdanau *et al.* (2014). The key idea was that instead of using a fixed representation,  $\mathbf{c}$ , from the encoder, compute a new representation at every timestep. This representation would be computed by *attending* to those tokens from the input sequence which are more important for generating the output word at time  $t$ . More specifically, the encoder

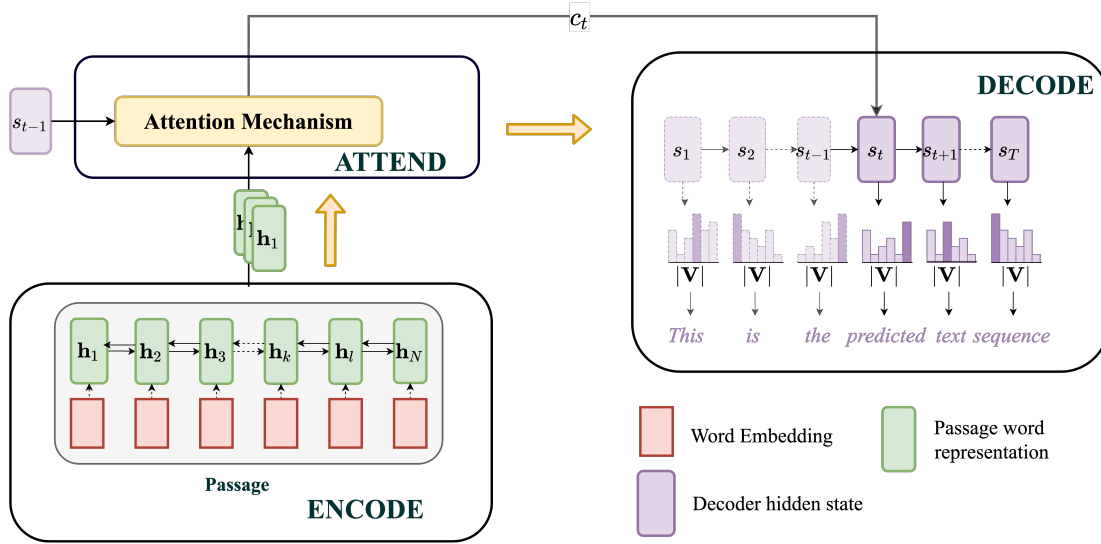


Figure 2.6: An RNN-based sequence to sequence model. It consists of encode, attend, and decode modules.

will compute a representation,  $\mathbf{h}_t$  for every token, just as explained earlier. The attention network will then compute a new context vector,  $\mathbf{c}_t$ , at every timestep  $t$  as a weighted combination of contextual representations of all the words in the input sequence, *i.e.*,  $\mathbf{H}$ , as shown below:

$$\mathbf{c}_t = \sum_{i=1}^N \alpha_{t,i} \mathbf{h}_i \quad (2.25)$$

$$a_{t,i} = \mathbf{v}_a^T \cdot \tanh(\mathbf{W}_a \cdot [\mathbf{s}_t : \mathbf{h}_i]) \quad (2.26)$$

$$\alpha_{t,i} = \frac{\exp(a_{t,i})}{\sum_{i=1}^N \exp(a_{t,i})} \quad (2.27)$$

Here,  $\mathbf{v}_a \in \mathbb{R}^{l_1}$ ,  $\mathbf{W}_a \in \mathbb{R}^{l_1 \times 2l}$  are learnable parameters. This context vector is then fed to the decoder LSTM as shown below.

$$\tilde{\mathbf{s}}_t, \mathbf{s}_t = \text{LSTM}(\mathbf{s}_{t-1}, [\mathbf{e}(\tilde{y}_{t-1}) : \mathbf{c}_t]) \quad (2.28)$$

The output  $y_t$  is a probability distribution over the words in the vocabulary. Figure 2.6 illustrates the complete architecture for one such network with the three modules, *viz.*, encoder, attention and decoder.



## 2.7 Evaluation Metrics

We now discuss the various evaluation metrics used for evaluating different NLG models proposed in this work.

### 2.7.1 BLEU

BLEU (Papineni *et al.*, 2002) was one of the initial metrics proposed to automate the evaluation of Machine Translation systems. It is a precision-based metric, which evaluates the generated hypotheses against the given references based on  $n$ -gram matches. In particular, it uses a modified precision defined as follows:

$$p_n = \frac{\sum_{H \in \text{hypotheses}} \sum_{n\text{-gram} \in H} \text{Count}_{clip}(n\text{-gram match with reference})}{\sum_{H \in \text{hypotheses}} \sum_{n\text{-gram} \in H} \text{Count}(n\text{-gram})} \quad (2.29)$$

The above definition ensures that incoherent sequences such as “*what what what?*” do not get unduly rewarded if the reference sequence contains the word “*what*” only once. This is done by clipping the number of matches between the hypotheses and the reference sentence by the number of occurrences of the corresponding  $n$ -gram in the reference. Thus, for the above example, the modified 1-gram precision would be 0.33 and not 1.0. Another limitation with a precision based metric is that it could reward a shorter hypothesis. For example, if the system only generates “*what*”, the modified precision will be 1.0, stating that it is a very good question. To mitigate this, the authors introduce a brevity penalty as given below:

$$BP = \begin{cases} 1, & \text{if } h > r \\ e^{(1-\frac{r}{h})}, & \text{if } h \leq r \end{cases} \quad (2.30)$$

The default definition of precision inherently penalizes hypotheses that are longer than the given reference. Thus, the brevity penalty is set to 1 for such cases. However, if the hypothesis is shorter than the reference, then the brevity penalty is set to a value as defined in Equation 2.30. Based on the modified precision and the brevity penalty, the

BLEU score is then defined as follows:

$$\text{BLEU-N} = BP \cdot \exp \left( \sum_{n=1}^N w_n \cdot p_n \right) \quad (2.31)$$

$$\equiv \exp \left\{ \sum_{n=1}^N w_n \cdot p_n + \max \left( 0, \frac{|r|}{|h|} - 1 \right) \right\} \quad (2.32)$$

BLEU computes the weighted geometric mean of the precision across all the  $n$ -grams up to  $N$ . BLEU-4 is typically used for evaluating NLG systems. Over the years, several studies (Callison-Burch *et al.*, 2006; Callison-Burch, 2009; Wieting *et al.*, 2019) have shown that BLEU correlates poorly with human judgments. Despite this, it is still one of the most widely used automated evaluation metrics for evaluating NLG systems.

## 2.7.2 NIST

The BLEU metric as defined above gives the same weightage to all the  $n$ -grams co-occurring in the hypothesis and the reference. However, some  $n$ -grams may be more informative than others. For instance, the trigram “*of the following*” is less informative than the trigram “*machines can learn*”. The NIST (Doddington, 2002) metric was proposed to capture the information gain of an  $n$ -gram. The information gain for a  $n$ -gram =  $\{w_1, \dots, w_n\}$  is defined as follows:

$$\text{Info}(w_1, \dots, w_n) = \frac{\# \text{ of occurrences of } w_1, \dots, w_{n-1}}{\# \text{ of occurrences of } w_1, \dots, w_n} \quad (2.33)$$

The authors further compute the arithmetic mean instead of the geometric mean to make the metric less sensitive to low co-occurrence counts for higher  $N$ . Further, they also change the brevity penalty not to take into account small variations in the hypotheses length. The final metric is thus defined as follows:

$$\text{NIST} = \sum_{n=1}^N \left\{ \frac{\sum_{\text{all } n\text{-grams that co-occur}} \text{Info}(w_1, \dots, w_n)}{\sum_{\text{all } n\text{-grams in hypotheses}} (1)} \right\} \exp(\beta \cdot \log^2(\min(\frac{|h|}{|r|}, 1))) \quad (2.34)$$

Note that  $\beta$  is set such that the brevity penalty is 0.5 when the average length of hypotheses is two-thirds of the average length of the reference text. Also, the default

value of  $N$  is set to 5.

### 2.7.3 METEOR

There are two major limitations of the BLEU and NIST metric as defined above. First, they only consider exact word match, and second, they only consider precision and do not take recall into account. To overcome these limitations, Lavie *et al.* (2004) proposed a new metric, METEOR, which also allows for partial matches (*e.g.*, the word *derive* in a hypothesis will be matched to the word *deriving* in the reference). These matched unigrams are then used to compute the F-Score instead of precision as used in BLEU and NIST.

$$P = \frac{\text{\#mapped\_unigrams}}{\text{\#words\_in\_hypotheses}} \quad (2.35)$$

$$R = \frac{\text{\#mapped\_unigrams}}{\text{\#words\_in\_reference}} \quad (2.36)$$

$$\text{F-Score} = \frac{\alpha P.R}{\alpha.P + (1 - \alpha).R} \quad (2.37)$$

Note that in the above form, this metric only considers unigrams. To check if there are larger  $n$ -gram matches they introduce a ‘fragmentation penalty’ as follows:

$$FP = 0.5 * \left[ \frac{\text{\#chunks}}{\text{\#mapped\_unigrams}} \right]^\delta \quad (2.38)$$

A chunk refers to a contiguous matching subsequence in the hypothesis and reference. It ranges from 1 to  $\text{\#mapped\_unigrams}$ , where it will be 1 if the reference and hypotheses are an exact match (word or stem), and  $\text{\#mapped\_unigrams}$  if none of the matched unigrams are in continuous order in the reference. Thus a larger chunk match leads to a lower ‘fragmentation penalty.’ The final score is computed as follows:

$$\text{METEOR Score} = \text{F-Score} * (1 - FP) \quad (2.39)$$

In this work, we use a variant of METEOR called METEOR-Universal (Denkowski and Lavie, 2014) which uses four different kinds of matchers, *viz.*, exact word matcher, stemmed word matcher, synonym matcher and paraphrase matcher. It assigns a different

weight to each of these matches. For example, if a word in the hypothesis exactly matches a word in the reference, it may get a higher weightage than when the stemmed form of a word in the hypothesis exactly matches the stemmed form of a word in the reference. Further, it gives a different weightage to content words and function words.

## 2.7.4 ROUGE

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin, 2004) is a set of evaluation metrics that were proposed in the context of automatic summarization. In contrast to BLEU, which is a precision-based metric, this metric is a recall based metric and is computed as follows:

$$\text{ROUGE-n} = \frac{\sum_{r \in \text{references}} \sum_{\text{n-gram} \in R} \text{Count}(\text{n-gram match with hypotheses})}{\sum_{\text{all n-grams in references}} (1)} \quad (2.40)$$

ROUGE-1 (unigram) and ROUGE-2 (bigram) are commonly used in the literature. Another variant of ROUGE that is widely used is ROUGE-L, which is the F-measure computed based on the Longest Common Subsequence (LCS) between the hypothesis and the reference. Given two sequences, a common subsequence is the set of words in both the sequences in the same order, but unlike  $n$ -grams, the common subsequence does not need to be contiguous. LCS is the longest of such common subsequences. For example, given the hypothesis, “*the boy went home*” and the reference, “*the boy will go home*”, “*the boy home*” is the longest common subsequence even though it is not contiguous. More formally, ROUGE-L is computed as follows:

$$P_{lcs} = \frac{|LCS(h, r)|}{\#words\_in\_hypotheses} \quad (2.41)$$

$$R_{lcs} = \frac{|LCS(h, r)|}{\#words\_in\_reference} \quad (2.42)$$

$$\text{ROUGE-L} = \frac{(1 + \beta^2)P_{lcs}R_{lcs}}{R_{lcs} + \beta^2P_{lcs}} \quad (2.43)$$

## 2.8 Summary

In this chapter, we introduced the essential concepts needed to understand the work described in this thesis. In particular, we discussed the different tasks addressed in this work, *viz.*, query-based abstractive summarization, structured data to text generation and automatic question generation. We then gave a quick primer on artificial neural networks, recurrent neural networks, sequence to sequence models and attention mechanisms. Finally, we discussed the different metrics used for evaluating the models proposed in this work.

## CHAPTER 3

### Related Work

The work done as part of this thesis was during the third phase of the evolution of NLG when RNN-based seq2seq models became very prominent and were being adopted for several NLG tasks. In parallel, various works were identifying numerous limitations in these models. Some of the limitations were seen across tasks, *i.e.*, task-agnostic limitations such as repeating phrases, and other limitations were task-specific such as generated questions being unanswerable in AQG. Several strategies (*e.g.*, additional auxiliary losses, architectural changes) were being proposed to overcome such limitations or (and) boost the performance of such systems.

We focused on different *tasks*: query-based abstractive summarization (QBAS), structured data to text generation (SD2T), and automatic question generation (AQG). We identified the following *limitations*: i) repeating phrases, ii) inadequate transition among fields in SD2T, ii) limited answerability of generated questions in AQG. Lastly, we proposed different *strategies* for RNN-based seq2seq models to mitigate these limitations: i) diversifying context vectors, ii) providing explicit reward signals to improve upon task-specific aspects, and iii) generating output in multiple passes. We present a pictorial representation of where our works are placed in view of the related works specifically for RNN-based seq2seq models. For instance, for our work (Nema *et al.*, 2017) we can observe that is proposed for **QBAS** task that focuses on **avoiding repeating phrases** by *diversifying the context vectors*. The other works on **avoiding repeating phrases** follow a different strategy to tackle this limitation.

We organize the related works based on the following three questions: (i) *What are state of the art models for different tasks considered in this thesis?* (ii) *What are the related works that propose solutions to similar limitations as identified in this thesis?* (iii) *What are the related works which have proposed the same techniques in a different context? (e.g., what are the works that have used the idea of diversifying context vectors but not necessarily for avoiding the problem of repeating phrases?* The rest of the

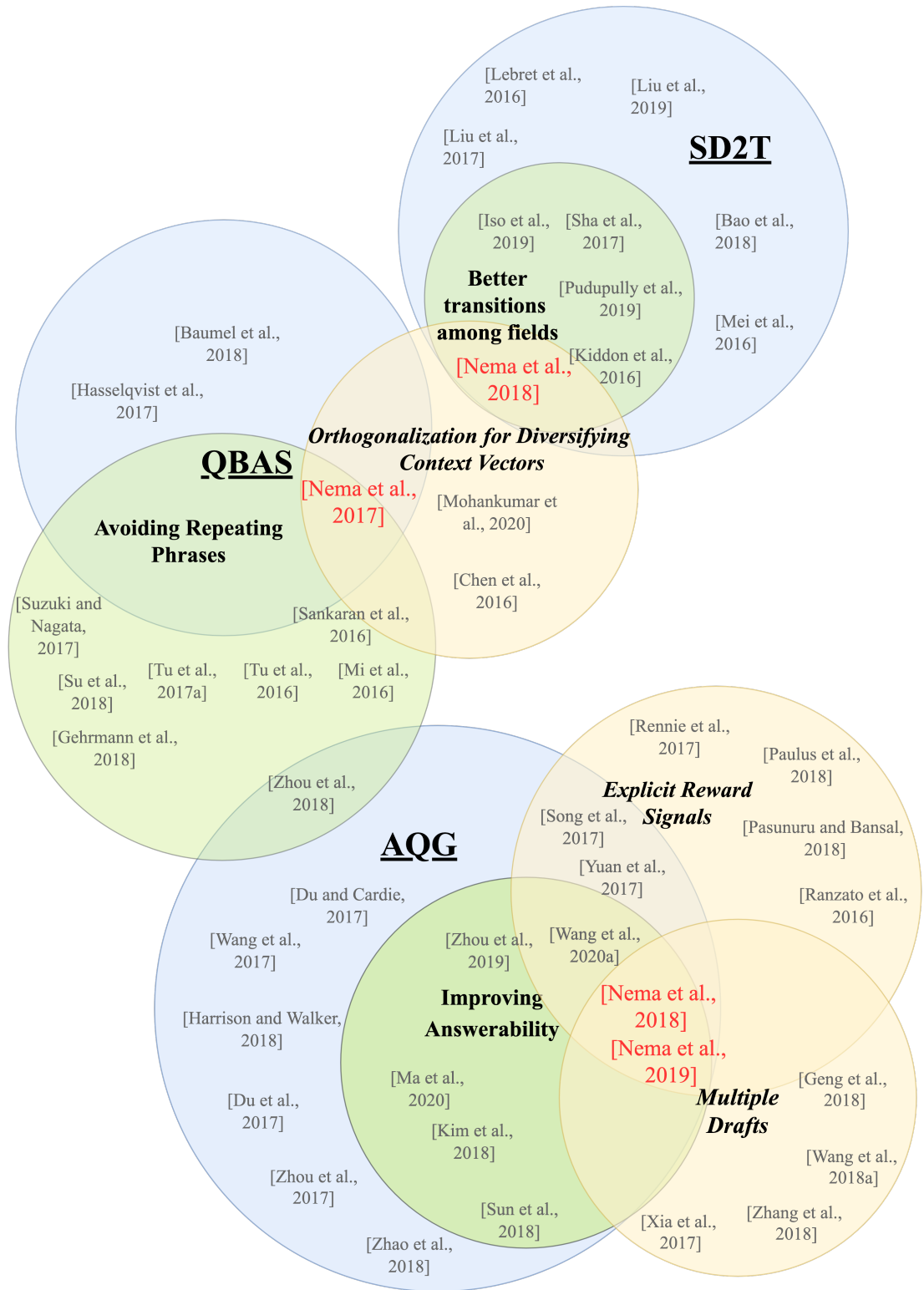


Figure 3.1: An overview of the existing works. across three dimensions: **Tasks**: QBAS (Query based Abstractive Summarization); AQQ (Automatic Question Generation); SD2T (Structured Data to Text Generation); **Limitations**: Avoiding Repeating Phrases, Better Transitions among fields, Improving Answerability; **Strategies**. We highlight the works based on this thesis in red. Note that the enlisted works are specific to RNN-based seq2seq models.

chapter is organized as follows. In Section 3.1.1, we give a brief overview of the models proposed for the tasks relevant to this thesis. In Section 3.2, we discuss the relevant works that focus on the same limitations of existing models as identified in this thesis. In Section 3.3, we highlight related works that use similar strategies to enhance seq2seq models.

### 3.1 State of the art models for different tasks considered in this thesis

This section gives a broad overview of various models proposed across years for the SD2T, AQG, and QBAS tasks. For each task, we present a brief history of models proposed in the pre-deep learning era and then give a detailed overview of works based on deep learning techniques, specifically the relevant works based on RNN-based seq2seq models.

#### 3.1.1 Task: Structured data-to-text generation

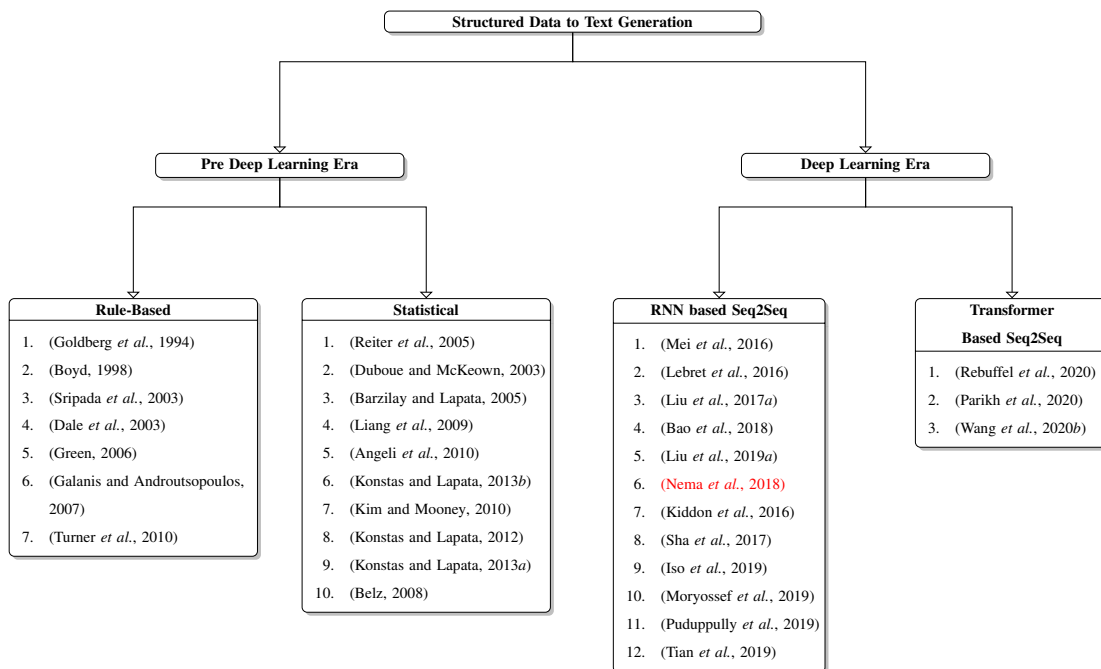


Figure 3.2: Taxonomy of models proposed for the task of Structured Data to Text Generation

Structured data to text generation (SD2T) broadly consists of three phases: i) *con-*



*tent selection*: selecting the information from the structured data that should be used to generate the output, ii) *sentence planning*: deciding what should be generated in various segments of the output sequence, iii) *surface realization*: actually writing each segment and fusing them. This task has applicability in several domains: writing weather reports (Goldberg *et al.*, 1994; Mei *et al.*, 2016), explaining biomedical reports to non-expert users (Green, 2006), writing selective descriptions from database (Elhadad and Robin, 1996), writing game summaries from game statistics (Wiseman *et al.*, 2017), and so on. Given its wide applicability, several systems have been developed since the early 1990s. Figure 3.4 shows the relevant literature for structured data to text generation. Before deep learning, models were primarily rule-based or statistical-based. Statistical-based models used data-driven approaches for one or more of the mentioned phases. With deep learning, RNN-based and Transformer-based seq2seq models are being widely adopted for SD2T. We describe some works from each of the mentioned categories below.

## **Pre Deep Learning Era**

*Rule-Based Approaches*: The Forecast Generator (FoG) system (Goldberg *et al.*, 1994) was one of the early automated weather report systems. The rules for content selection were written by domain experts (meteorologists). Although the initial system modeled some paraphrase variations at the surface realization stage, domain experts preferred direct mapping to control the generated descriptions. Boyd (1998) generated descriptions for time-series data. It primarily focused on content selection, which was mainly governed by scale-scape theory used in image processing. The authors represented the content selected as attribute-value pairs, which they passed to a highly complex rule-based system: SURGE (Systemic Unification Realization Grammar of English) (Elhadad and Robin, 1996) for surface realization. The SUMTIME-MOUSAM (Sripada *et al.*, 2003) system for generating weather reports uses data-analysis strategies for content selection and then uses hand-crafted rules for each of the identified attributes from the data to generate the reports. Dale *et al.* (2003) focused on generating descriptions for way-routing systems. The authors used a path-based segmentation technique to identify meaningful segments based on turns, road status, and path length of a route descrip-

tion. These segments were again combined using hand-crafted rules to generate the final description. Similarly, rule-based systems in other domains (Green, 2006; Galanis and Androutsopoulos, 2007; Turner *et al.*, 2010) focused more on content selection and resort to hand-crafted rules for surface realization.

*Statistical Approaches:* Reiter *et al.* (2005) focused on identifying the right set of words to describe the segments. Through careful analysis of human written reports, the authors found that the word choice depended on the writers' preferences and linguistic context. Therefore, the authors used semantic features, writers' context, and temporal features to construct a decision tree for choosing the appropriate phrases.

For content selection, Duboue and McKeown (2003) first classify each record as *to-be-selected* or *not-to-be-selected*. They ensure the overall coherency of the description by taking some discourse features as an input to the classifier. Barzilay and Lapata (2005) generated descriptions for sports domain from game statistics. Here, the content selection was a statistical model based on decision trees. In contrast to the above method, all the records were collectively classified while optimizing individual label assignments and pairwise relations. Liang *et al.* (2009) built on this work and modeled the surface realization component as a Hierarchical Hidden Markov Model. Angeli *et al.* (2010) broke the generation into macro planning (identifying the records to be described), micro-planning (selecting the features specific to the record to be summarised), and then surface realization using the model proposed in Liang *et al.* (2009).

Konstas and Lapata (2013a) used a probabilistic context-free grammar to represent the structured data. To this end, they enlisted hand-crafted rules to represent the structured data in the form of hyper-graphs. The surface realization was then based on identifying the correct hypergraph and generating the sentence using dynamic programming. Similarly, Belz (2008) used a probabilistic context-free grammar for the surface realization stage while generating weather reports. Konstas and Lapata (2013b) point out that in their previous work (Konstas and Lapata, 2013a), they do not explicitly model ordering of the selected content, which could lead to incoherent generations. The authors, therefore, extend their model by additionally learning discourse grammar based on Rhetorical Structure Theory (Mann and Thompson, 1987).

## Deep Learning Era

*RNN based seq2seq models:* RNN based seq2seq models were initially designed for text-to-text generation. As the input in these cases is a simple word sequence and is unstructured, the same architecture may not be suitable for structured data to text generation. Moreover, even if structured data is converted into a stream of words, the resulting generation will be sub-optimal, as the model cannot exploit the inherent structure in the input data. Various works (Mei *et al.*, 2016; Lebret *et al.*, 2016; Liu *et al.*, 2017a; Bao *et al.*, 2018; Liu *et al.*, 2019a) that have adopted seq2seq models for structured data to text generation have first focused on incorporating the structure and then optimized it for content selection and sentence planning stages.

Mei *et al.* (2016) was one of the first works to incorporate RNN based seq2seq models for generating text from structured data. The authors focused on generating weather reports using WEATHERGOV dataset and game summaries using ROBOCUP dataset. They used an LSTM based encode module to compute representations for each record. The authors observed that only a few records were used to write the description. Therefore, the authors used a coarse-to-fine attention mechanism that first estimates the probability of a record being selected and then determines a record being selected based on the intermediate generation state or the decoder state. These two attention distributions are then used to select the records that should be described in the output.

Earlier models were largely studied in the context of small datasets such as WEATHERGOV (Liang *et al.*, 2009), ROBOCUP (Chen and Mooney, 2008), NFL RECAPS (Barzilay and Lapata, 2005), PRODIGY-METEO (Belz and Kow, 2009) and TUNA Challenge (Gatt and Belz, 2010). Unlike the datasets mentioned above, Lebret *et al.* (2016) proposed a new dataset to create one-line descriptions from Wikipedia Infoboxes consisting of 700K samples. Similar to Mei *et al.* (2016), the authors generated the descriptions conditioned on local characteristics and global characteristics. The former represented the previously generated words (local context) that were carefully encoded based on the records to which these words belonged. The latter referred to the overall infobox representation consisting of all the rows. Here, the authors generated the descriptions using a feedforward neural network.

Liu *et al.* (2017a) proposed a novel variant of the LSTM with an additional field gate to encode the structure of the table in the token-level encoder. While encoding the given table, which is a sequence of  $\{field, token\}$  sets, they utilize the field gate that learns how much field information for the corresponding token should be encoded in the LSTM cell’s memory state. To represent a token in the context of a field, they used a similar embedding model as proposed in Lebret *et al.* (2016). Also, the authors used hierarchical attention at field level and word level to model global and local characteristics while generating the description. This dual attention strategy is also used in Bao *et al.* (2018).

Liu *et al.* (2019a) used a hierarchical encoder for encoding the hierarchy between the fields and the entities in it. Also, the authors use several auxiliary losses to improve the rendered descriptions. They state that the encoder module may not always capture the fields effectively. Therefore, they design a feedforward neural network based multi-label classifier that aims at classifying the fields based on the representations learned by the encoder. The authors use another text-encoder module that learns a meaningful representation for the ground-truth description. The other auxiliary task is to predict the sequence from this one representation. The above two auxiliary tasks are jointly trained.

*Transformer-Based Models:* Following our work (Nema *et al.*, 2018), several other advancements have been made in the field (Vaswani *et al.*, 2017; Radford *et al.*, 2019; Brown *et al.*, 2020; Yang *et al.*, 2019). For instance, following the success of transformer-based models, several models are now adopting transformer-based models (Wang *et al.*, 2020b; Rebuffel *et al.*, 2020; Parikh *et al.*, 2020; Xiao and Wang, 2021) for rendering descriptions from structured data. Wang *et al.* (2020b) focuses on generating faithful biography descriptions by adopting transformer based models and adding two content-matching constraints. The first one ensures that the latent representation of the generated text is consistent with the structured data. The second one constrains keywords from the input and the generated description to be identical. Rebuffel *et al.* (2020) proposed a hierarchical Transformer-based seq2seq model to encode the structure of the given input. In parallel, some works (Wang *et al.*, 2020b; Parikh *et al.*, 2020; Chen *et al.*, 2020; Xiao and Wang, 2021) have observed that the generated descriptions are

often not faithful to the given structured data. The authors hypothesized that this is because the training dataset is noisy. To this end, Parikh *et al.* (2020); Chen *et al.* (2020) propose high-quality datasets as benchmarks to evaluate if the models generate faithful descriptions. Moreover, Parikh *et al.* (2020) also highlights that even with the proposed high-quality training dataset, BERT-based sequence to sequence models still generate unfaithful descriptions. Note that these transformer based models were developed after our work (Nema *et al.*, 2018) and hence we do not compare with these models in our work.

### 3.1.2 Automatic Question Generation

AQG systems are based on the two fundamental aspects of asking a question: *what to ask* and *how to ask*. The former corresponds to *content selection*, and the latter corresponds to *question construction*. For a given input, the content selection module selects a question-worthy topic. In the question construction phase, the model generates a question to this specific topic using the given context. Before deep learning, the two modules were designed independently and primarily followed a rule-based approach. With the advent of deep learning, AQG systems were trained using an end-to-end architecture. We give an overview of the models proposed in both the phases below.

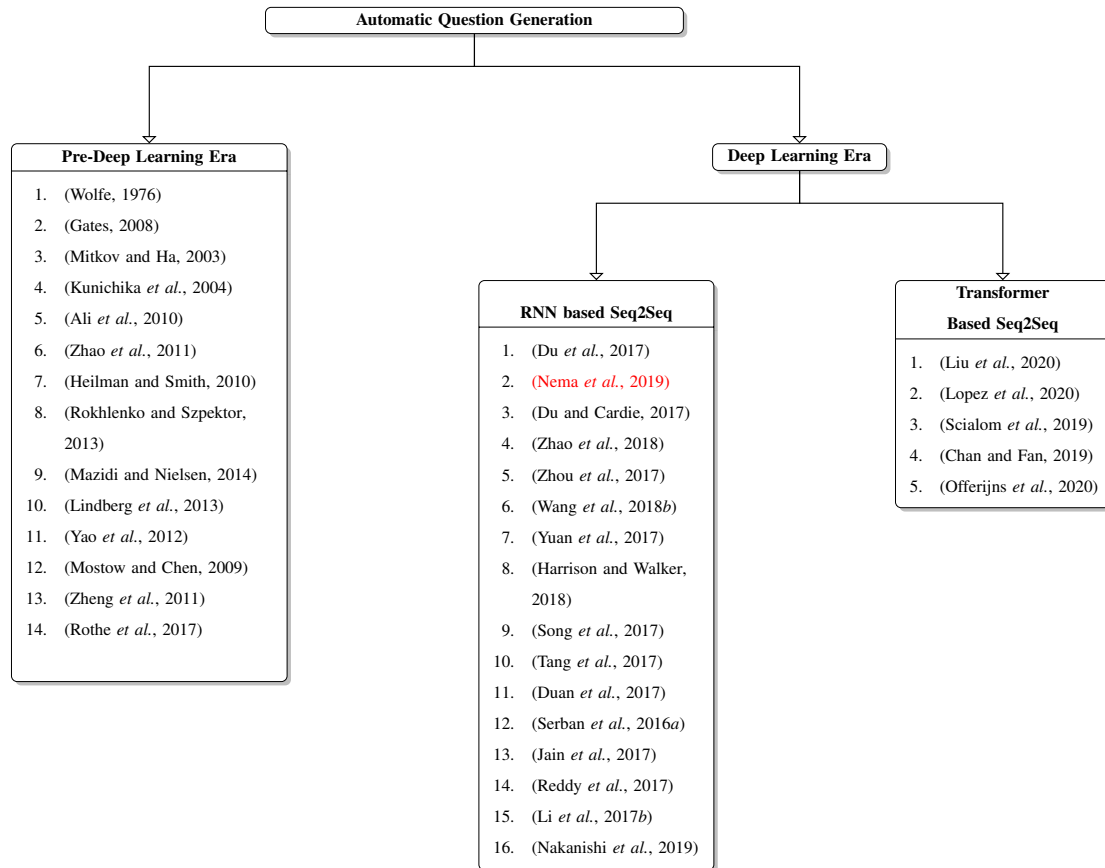


Figure 3.3: Taxonomy of models proposed for the task of Automatic Question Generation

## Pre-Deep Learning Era

In earlier models, the question construction phase was either transformation-based or template-based. In the first one, a declarative sentence is transformed into an interrogative form using a rule-based approach. In template-based models, a pre-defined question template is given, and the appropriate slots in this template are filled with content selected from the sentence. We highlight some works in each category below.

- **Transformation Based:** Various works (Wolfe, 1976; Gates, 2008; Ali *et al.*, 2010; Kalady *et al.*, 2010; Heilman and Smith, 2010; Varga and Ha, 2010) simplified the given sentence, and then syntactically parsed it to identify potential answer phrases. For each identified answer phrase, a corresponding question was generated by transforming the simplified sentence into a question through a set of hand-crafted rules. Wolfe (1976) was one of the early works proposed for AQG. The system generated 50 (question, answer) pairs for the reading comprehen-

sion task following the above-mentioned approach. Heilman and Smith (2010); Gates (2008) used Stanford’s T Surgeon tool (Levy and Andrew, 2006) to transform a declarative sentence to a question. As the transformations could also lead to unacceptable questions, Heilman and Smith (2010) further ranked the generated questions through a logistic regression model that classified a question into *{acceptable, unacceptable}* classes, based on question specific features such as length features, grammatical features, negation, *etc.* Some other works (Mannem *et al.*, 2010; Mazidi and Nielsen, 2014; Yao *et al.*, 2012) simplified the given sentence and then used semantic role labeling to identify predicates in the sentence about which a question can be generated.

- **Template Based:** Several works (Lindberg *et al.*, 2013; Mostow and Chen, 2009; Rokhlenko and Szpektor, 2013; Liu *et al.*, 2010) generated questions from predefined question templates. Labutov *et al.* (2015) collected high-level question templates through a crowd-sourcing pipeline. In parallel to generating questions from sentences, there was ongoing research on converting a given web query into a question. This task was first proposed in Lin (2008). Follow-up works (Zhao *et al.*, 2011; Zheng *et al.*, 2011) proposed template-based approaches for this task, where the templates were learnt from a large corpus consisting of web-query and corresponding question logs.

In the above works, we can observe that significant human effort was required in designing the rules, templates, and features for both content selection and question construction stages. The following deep learning based methods alleviated this problem.

## Deep Learning Era

*RNN-Based Seq2Seq Models:* Seq2seq based end-to-end architectures were soon proposed for question generation (Du *et al.*, 2017) after their success in other NLG tasks. As this task is complementary to the task of QA, several datasets such as SQuAD (Rajpurkar *et al.*, 2016), DROP (Dua *et al.*, 2019), HOTPOTQA (Yang *et al.*, 2018), MS-Marco (Nguyen *et al.*, 2016), *etc.*, which were originally proposed for QA, also fuelled the development of seq2seq based AQG systems.

1. **Generating Questions given a sentence**: Du *et al.* (2017) was the first work to propose a seq2seq based model for generating questions from a given sentence and its corresponding passage. The authors encode the sentence and the passage using a bidirectional-LSTM. They also attend to the relevant sentence words while generating the question. Note that the passage encodes the context information and is only used while initializing the decoder state. Du and Cardie (2017) built on top of this work and proposed a preliminary step of first identifying the question worthy sentences from a paragraph via a feedforward neural network-based classifier. To train this classifier, the authors labeled sentences consisting of more than one answer-span as question-worthy. The sentences predicted as question-worthy were then used to generate a question using the model proposed in Du *et al.* (2017).
2. **Generating Questions given a (sentence,answer) pair**: Zhou *et al.* (2017); Harrison and Walker (2018) passed the (sentence, answer) pair as an input, whereas in previous approaches, the AQG system itself extracted a potential answer. They proposed an RNN-based seq2seq model to generate questions assuming that the answer is a span in the given input. They introduced a feature-rich encoder that reads the sentence words and answer position, which helps the encoder locate the answer while reading the sentence. It also uses other linguistic features such as POS, NER tags to encode linguistic information. Note that only the sentence containing the answer span is passed as an input. Also, Harrison and Walker (2018) encode coreference labels from the passage to which the sentence belongs to enrich the sentence representation further. The authors also introduced a question encoder trained using a reconstruction framework with only target questions to learn meaningful question embeddings. The representation from the sentence encoder is constrained to be similar to this target question embedding.

Wang *et al.* (2018b) also generate questions specific to the given answer, but they pass the passage containing the answer as an input. They also encode the passage using the answer position feature and linguistic features and use a bidirectional LSTM network to encode the passage.

Yuan *et al.* (2017) also generated questions given (passage, answer) pairs, assum-



ing the answer span is present in the given input. The authors encoded the answer information at two stages. First, similar to the above work (Zhou *et al.*, 2017), they utilize the answer position feature while reading the given text. Second, they extract the passage words’ hidden representations that correspond to the answer span. They concatenate this representation with the answer words’ embeddings and feed it to another LSTM network. The final answer word representation is then utilized while attending to the passage and computing the decoder’s hidden state. Zhao *et al.* (2018) observed that passing richer context as passage led to performance degradation instead of boosting it. To mitigate this, the authors proposed an additional self-attention layer in the encoder to capture the intra-passage interaction.

S. No	Work	Answer-Aware	Features			Dataset
			CM	PL	LF	
1.	(Du <i>et al.</i> , 2017)	No		*		SQuAD
2.	(Du and Cardie, 2017)	No		*		SQuAD
3.	(Zhou <i>et al.</i> , 2017)	position-encoding	*		*	SQuAD
4.	(Harrison and Walker, 2018)	position-encoding	*	*	*	SQuAD
5.	(Yuan <i>et al.</i> , 2017)	answer-encoder	*	*		SQuAD
6.	(Wang <i>et al.</i> , 2018b)	position-encoding	*	*	*	SQuAD
7.	(Zhao <i>et al.</i> , 2018)	position-encoding	*	*		SQuAD, MS-MARCO
8.	(Nema <i>et al.</i> , 2019)	answer-encoder	*	*		SQuAD, DROP, HOTPOTQA

Table 3.1: Existing RNN-based seq2seq models for question generation given a text. Such models broadly consist of answer-aware module, and other features such as **CM**: Copying Mechanism, **PL**: Paragraph Level Information, **LF**: Linguistic Features. The last column enumerates the datasets used to validate the proposed model.

In Table 3.1 we give a concise comparison of the works discussed till now. We observe that several works (including ours) (Zhou *et al.*, 2017; Harrison and Walker, 2018; Yuan *et al.*, 2017; Wang *et al.*, 2018b; Zhao *et al.*, 2018; Nema *et al.*, 2019) had adopted a copying mechanism to overcome the problem of producing rare words which may not be in the decoder’s vocabulary. Some works Zhou *et al.* (2017); Harrison and Walker (2018); Wang *et al.* (2018b) have also encoded linguistic information for better question generation. Also, SQuAD is the most popular dataset used by all the AQG systems mentioned above.

### 3. QA and QG as dual tasks: Several works (Song *et al.*, 2017; Tang *et al.*, 2017;

Sachan and Xing, 2018) take the view that Question Generation (QG) and Question Answering (QA) are complementary tasks. For example, Song *et al.* (2017) proposed a unified model for QA and QG tasks. The proposed model consisted of passage and query encoders, and the interaction between the two sequences was captured from multiple perspectives (Wang *et al.*, 2017). This fused representation was then used to generate the output. Note that the network was trained separately for QG and QA tasks, although the architecture was the same. Similarly, Tang *et al.* (2017) propose RNN-based models for QA and QG tasks and train them jointly with the constraint that the joint probability of a correct (question, answer) pair estimated by the QA and QG models should be similar. Sachan and Xing (2018) also do a joint training for QA and QG tasks, using self-training algorithms.

4. **Generating Questions from different modalities:** Apart from generating questions from a passage, several other works focus on generating questions from a different context: Knowledge-Graph (Serban *et al.*, 2016a; Reddy *et al.*, 2017), images (Jain *et al.*, 2017; Li *et al.*, 2017b), dialogs (Nakanishi *et al.*, 2019), *etc.*

*Transformer Based Models:* Similar to other NLG tasks, transformer-based models are being adopted for QG tasks as well. Lopez *et al.* (2020) finetune GPT-2 language model (Radford *et al.*, 2019) with SQuAD and RACE datasets. Scialom *et al.* (2019) adopted Transformers for QG where it did not constrain the model to generate questions specific to a given answer. Also, the authors integrated the copynet mechanism in vanilla transformer models to mitigate the rare word problem. Chan and Fan (2019) adopted BERT-based model for AQG. As the BERT model does not have a text generation module, the authors modify the mechanism to generate a question. First, the representations of the sentence and the answer are concatenated to generate an initial state based on which the first word is generated. The representation of the generated word is then concatenated with the representations of the sentence and the answer, which is then used to generate the next word, and so on. Offerijns *et al.* (2020) focused on generating distractors for multiple-choice questions, which used GPT-2 model.

Based on the above discussion, we can infer that AQG is a growing field, and apart from generating questions given a context, several works are now focusing on gener-

ating clarification questions (Xu *et al.*, 2019), generating question-answer pairs (Lee *et al.*, 2020), unanswerable questions (Zhu *et al.*, 2019) using deep learning based models.

### 3.1.3 Query Based Abstractive Summarization

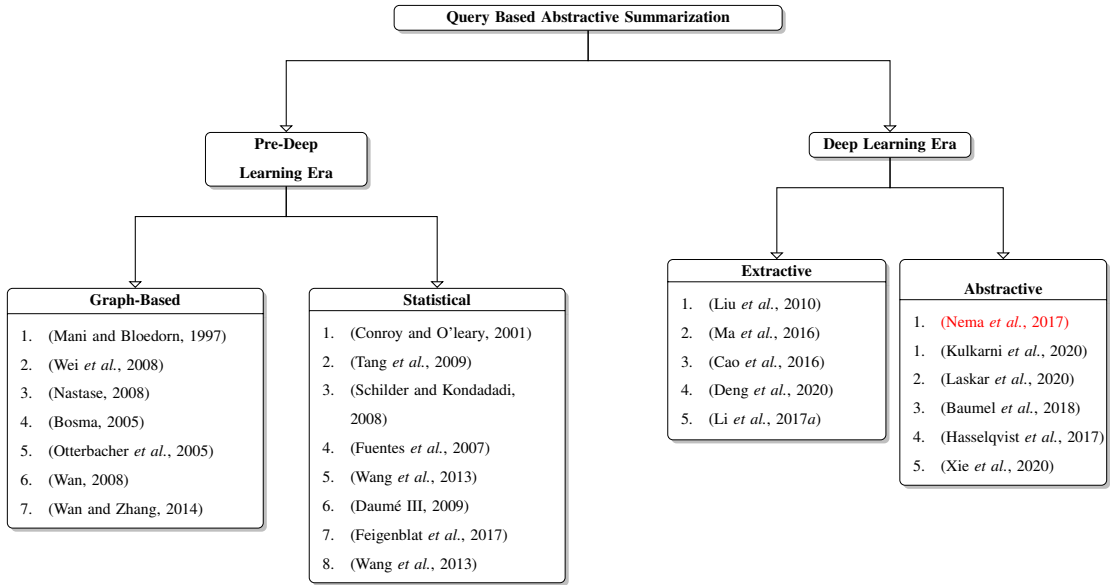


Figure 3.4: Taxonomy of models proposed for the task of Query Based Abstractive Summarization

Automatic Text Summarization systems have been explored since the early 1950s (Luhn (1958); Edmundson (1969)). These works can be classified across 3 axes (i) *extractive* versus *abstractive* summarization, (ii) *generic* versus *query-based* summarization, (iii) *single document* versus *multi document*. In extractive summarization, the goal is to identify salient sentences in the document that best represent its central idea. On the other hand, abstractive summarization refers to writing the summary from scratch based on the inferences drawn from the document. Next, in *generic* summarization, the summary corresponds to the central idea of the document, whereas in *query-based* summarization, the summary is specific to a given query. Lastly, in *single document* summarisation, the summary is generated from a single document, whereas in *multiple document summarisation*, the summary is generated from multiple documents. In this thesis, we primarily focus on query-based abstractive summarization for a single document, and we briefly discuss the literature specific to it.

## Pre Deep Learning Era

Earlier models focused on generating query (topic) based extractive summaries from multiple documents. The approaches mainly were graph-based or statistical-based. We briefly describe some works for both the approaches below.

*Graph-based Models:* One of the early works (Mani and Bloedorn, 1997) focused on generating extractive summaries for multiple documents for a given topic or perspective. The authors first created a graph by detecting relevant entities and relations using information extraction techniques. The graph was then traversed using spreading activation algorithm through which topic-relevant nodes were identified. The authors then detected the sentences that best covered such nodes, and these sentences were then ranked using TF-IDF scores. Note that the above model used the query to identify only the salient nodes in the graph. However, Wei *et al.* (2008) showed that incorporating query in measuring sentence-sentence similarity boosted the performance. Nastase (2008) first expanded the query using external knowledge sources such as Wikipedia and then used spreading activation algorithm to identify the relevant sentences. Bosma (2005) used Rhetorical Structure Theory (Mann and Thompson, 1987) to convert the discourse structure of the text into a graph, where the relevant sentences are identified using a graph search algorithm. Otterbacher *et al.* (2005) proposed a Lexrank method, which represented text as a graph of passages, where the edges were weighted based on their pairwise lexical similarity. The authors first identified the passages relevant to the query using TF-IDF scores and then performed a random walk on this graph to recursively extract similar passages. Similarly, Wan (2008) proposed a topic-sensitive Pagerank algorithm that performs random walk while taking sentence relevance with respect to the topic into account. Wan and Zhang (2014) use information certainty to identify relevant sentences while summarizing news articles for a given event/topic. The authors first predict the certainty scores using the Support Vector Regression (SVR) model for each sentence in the article and then incorporate certainty scores in the transition probability matrix, which is fed to a graph-based algorithm to identify relevant sentences.

*Statistical Models:* Here, we first discuss some approaches which use generative models.

Generative models: Conroy and O’leary (2001) proposed a Hidden Markov model for query-based summarization. The authors first extracted the relevant sentences based on surface-level features and query terms and then used a pivoted QR algorithm to remove redundancy from the selected sentences. Tang *et al.* (2009) observed that documents are often a mixture of topics and could be related to multiple topics mentioned in the query. Therefore, they proposed a query-based Latent Dirichlet Allocation (LDA) model, which estimated a mixture of a document-specific topic distribution and a query-specific topic distribution. An importance score for each sentence was computed using the learnt topic model, and the most salient sentences were extracted as a summary. Daumé III (2009) proposed BayesSum model that focuses on the following question: “why a particular document is relevant, and others are not ?” The authors deploy a Bayesian Statistical Model, where they hypothesize that each word in the document generated from a mixture of these three language models: i) an English filler word (English language model), ii) relevant to the query (query-specific language model), iii) depicts some background information about the document which is irrelevant to the query (document model). A saliency score is then assigned to each sentence based on the learnt generative model to rank the sentences.

Other approaches: Schilder and Kondadadi (2008) propose a fast summarization technique where the authors use only word frequency-based features of document clusters, given query and query description. The authors then extracted relevant sentences through an SVR model based on the derived word-frequency features. Fuentes *et al.* (2007) used SVM to select the relevant sentences, where sentence-based features were designed to capture (i) the similarity between sentence and query, (ii) cohesion between the sentence and other sentences, and (iii) surface-level information about a sentence. Feigenblat *et al.* (2017) posed the extraction of relevant sentences as a constrained global optimization problem. The authors proposed an unsupervised approach based on the cross-entropy method, where the objective was to maximize a quality target function. The quality target function was based on features such as the similarity between query and sentences, sentence length, the relevant position of the sentence in the document, *etc.* This work was extended in a follow-up work (Roitman *et al.*, 2020), where the authors decouple the process of identifying salient sentences and generating a topic-focused summary and instead proposed a cascaded optimization approach. Few

works (Wang *et al.*, 2013; Zajic *et al.*, 2006) focus on adopting sentence compression technique to condense the extracted salient sentences.

## Deep Learning Era

We describe some relevant Deep Learning based techniques for both extractive and abstractive query-based summarization below.

*Extractive Summarization:* Liu *et al.* (2012) proposed an unsupervised approach based on Restricted Boltzman machines. The model consisted of modules for (i) concept extraction: which extracts sentences relevant to the query, (ii) reconstruction validation: which reconstructs the document from the latent representation, and (iii) summary generation: which extracts the most informative segments in the shortlisted sentences using dynamic programming. The query is incorporated by penalizing the reconstruction loss more for words that match the query. Ma *et al.* (2016) pointed that the above methods did not take the global semantics of the documents into account. They proposed a reconstruction framework *DocRebuild* which used neural networks to compute a document representation using the whole document. Another document representation was constructed by selecting salient sentences as a summary. The goal was to select salient sentences to minimize the gap between these two representations.

Cao *et al.* (2016) modeled the task of extracting the relevant sentences using Convolutional Neural Networks (CNNs) with an attention mechanism. The query and sentence in the documents are embedded using CNNs. The attention network learns the relevance score for each sentence with respect to the query. These relevance scores are used to compute a document embedding which is a weighted aggregation of sentence embeddings. A sentence similar to the learnt document embedding is classified as a salient sentence.

Li *et al.* (2017a) introduced an unsupervised approach with a cascaded attention mechanism that estimates the topic-specific salient information from the given set of documents. The cascaded attention model is trained through a reconstruction framework. The authors add sparsity constraints so that condensed salient information is retrieved. Moreover, the authors use fine-grained and coarse-grained sentence com-

pression strategies to generate a compressed summary.

Singh *et al.* (2018) proposed an unsupervised approach to extract relevant sentences. The authors state that the previous unsupervised approaches identify the relevant sentences using short-range language models. They mitigate this shortcoming by proposing a novel variant of LSTMs with an additional memory state to track sentence-level information. The authors first use this to compute relevance ordering. The sentences are then selected using integer linear programming (ILP) approach to increase diversity and reduce redundancy.

Deng *et al.* (2020) proposed a hierarchical model to capture the interaction between the query and the document at word level and sentence level. Each sentence is sequentially labeled as being relevant to the query or not by passing this aggregated information through an LSTM network. Ya *et al.* (2020) also utilizes word and sentence level attention and bridges the gap between query and document using external knowledge.

*Abstractive Summarization:* Before our work, abstractive summarization approaches were mainly focused on summarizing the given document’s central idea. Therefore, we first highlight some of the abstractive summarization models that focus on generating generic summaries. We then briefly describe some of the works that followed our work for query-based abstractive summarization.

- **Generic Summarization:** Rush *et al.* (2015) was one of the early works that adopted the RNN-based seq2seq model for abstractive summarization, where the authors validated their model on the GigaWord and DUC corpus. Similarly, Lopyrev (2015) used neural networks to generate news headlines from short news stories. Chopra *et al.* (2016b) extended the work of Rush *et al.* (2015) and reported further improvements on the two datasets. Hu *et al.* (2015) introduced a dataset for Chinese short text summarization and evaluated a similar RNN encoder-decoder model on it. With the potential seq2seq models showcased for abstractive summarization, many models have been proposed to generate abstractive summaries. Some of the recent models use a wide variety of techniques such as (i) using hierarchical attention (Nallapati *et al.*, 2016), (ii) using a coverage-mechanism (See *et al.*, 2017; Suzuki and Nagata, 2017; Chen *et al.*, 2016), (iii)

training via explicit reward signals (Paulus *et al.*, 2018), (iv) using a module to cover all salient points (Gehrmann *et al.*, 2018), (v) combining extractive and abstractive strategies, and (vi) using a copy mechanism (Gu *et al.*, 2016; See *et al.*, 2017). Recently, transformer-based models (Liu and Lapata, 2019; Dong *et al.*, 2019; Lewis *et al.*, 2019) have replaced RNN-based seq2seq models for abstractive summarization as well.

- **Query-based Summarization**: In parallel to our work, Hasselqvist *et al.* (2017) also proposed a seq2seq model for query-based abstractive summarization. However, it used CNN/Dailymail dataset (Hermann *et al.*, 2015), a large-scale cloze-style question answering dataset on news articles. Each news had some short highlights associated with it, and each highlight was written for some entities. The authors mapped this dataset to QBAS as follows: the highlight was taken as a summary, the document was taken as it is, the entity in the highlight was marked as topic/query.

Baumel *et al.* (2018) point out that it is difficult to train a query-based multi-document summarization system efficiently due to the lack of a large-scale dataset. To this end, the authors used a pre-trained seq2seq model for generic summarization and augmented a query-relevance model. The authors first identify query-specific passages based on the unigram overlap between query and passages, TF-IDF scores, and word2vec encodings. The relevant passages are passed to the pre-trained seq2seq model. This procedure is repeated for other documents until the summary length does not exceed 250 words.

To tackle the lack of large-scale datasets for query-based multi-document summarization, Laskar *et al.* (2020) introduced a novel approach to train such systems using weakly-supervised learning. The proposed model consists of RoBERTa (Liu *et al.*, 2019b) and BERTSUM (Liu, 2019). RoBERTa is primarily used to augment document-level relevant sentences in summary along with the overall summary. BERTSUM, a BERT-based seq2seq model, is then fine-tuned using this augmented dataset, which then generates a summary. To address the lack of large-scale datasets, recently Kulkarni *et al.* (2020) used large-scale Question Answering datasets such as Google Natural Questions and Common Crawl to



automatically create a query-based multi-document dataset for extractive and abstractive summaries. Xie *et al.* (2020) also proposed a transformer-based architecture for query-based abstractive summarization, where the authors modified the computed self-attention in each layer based on the matching score between the query and a sentence. They validated the proposed model on the Debatespedia dataset proposed in our work and HOTPOT-QA, which was tweaked for query-based summarization.

## **3.2 Works addressing the same limitations in Seq2Seq models as identified in this thesis**

This section discusses related works that focus on addressing similar limitations of seq2seq models as considered in this thesis, *viz.*, i) avoiding repeating phrases, ii) better transitions among fields for structured data to text generation, and iii) improving answerability for automatic question generation.

### **3.2.1 Avoiding Repeating Phrases**

We further divide the existing works that address the problem of repeating phrases into two categories: explicit and implicit approaches. The explicit approaches are the ones where generating outputs with fewer repeating phrases is a stated goal of the work. This category has been further divided based on *how* repetitions have been alleviated. The majority of the relevant approaches mitigated repetitions by tracking the previously attended tokens. The other category refers to miscellaneous works that address this problem by using additional auxiliary losses or by using better decoding strategies. On the other hand, the implicit approaches refer to the models that focused on improving the overall generation quality, such as increasing fluency and/or adequacy of the generated text. The primary aim here was not to mitigate the repetitions, however as fluent/adequate descriptions will have fewer repetitions in general, this problem also gets alleviated to some extent. Some of the relevant works in each of the mentioned categories have been listed in Figure 3.5.

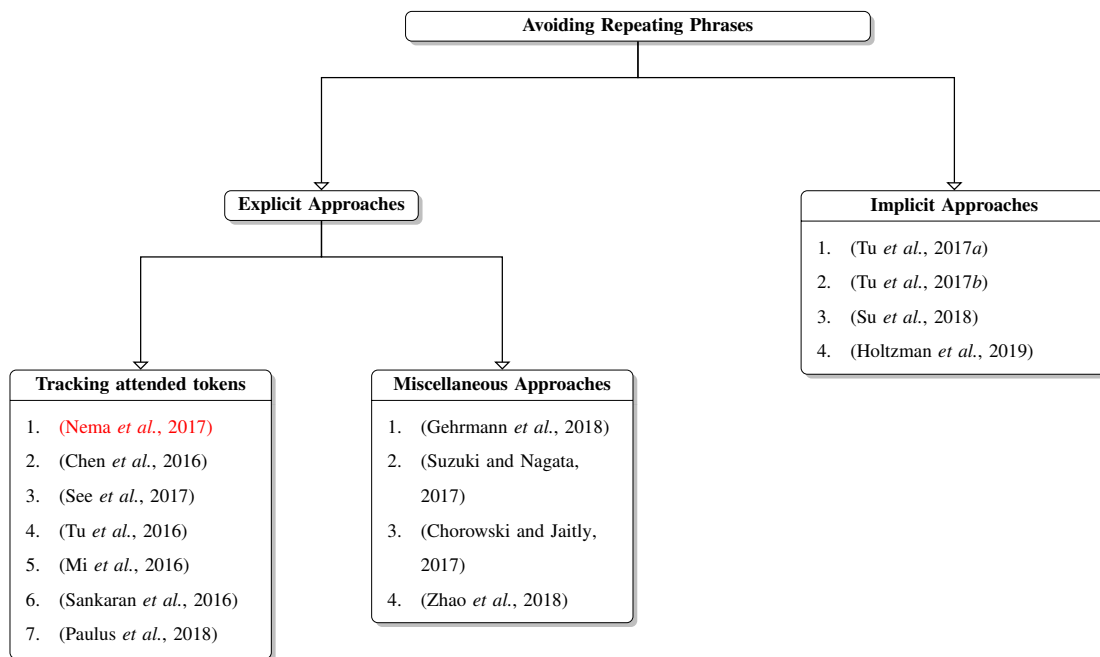


Figure 3.5: Taxonomy for works that avoid repeating phrases in the output

## Explicit Approaches

*Tracking attended tokens:* Several works (Tu *et al.*, 2016; Mi *et al.*, 2016; See *et al.*, 2017; Tu *et al.*, 2017b; Chen *et al.*, 2016) claimed that one of the primary reasons for incoherent generations in NLG systems is the inability of attention mechanisms to track what has been attended to in the previous decoder timesteps. Consequently, similar contexts or phrases may be attended to repeatedly. As the decoder module is conditioned on the attended input representation, this could lead to either similar words being predicted (repeating phrases) by the decoder module or missing some phrases in the target sequence corresponding to the unattended source phrases (limited coverage).

Tu *et al.* (2016) observed the issues mentioned above in Machine Translation. The authors hypothesized that limited coverage leads to either over-translation (repeating phrases) of some source words or under-translation (some source words being left out in the translated sentence). Therefore, they focused on increasing the coverage of the model. Their approach is motivated by the strategy used in statistical machine translation models to keep track of source words that have been generated (Koehn, 2004; Koehn *et al.*, 2003). The authors associated a learnable coverage scalar for each source word to track if it was entirely translated, *i.e.*, if the coverage value is close to 1. The coverage scalar is modeled using a GRU to keep track of the previous attention distri-

butions. These coverage values were also fed to the attention mechanism so that in the attention distribution predicted at the current step, the source words attended to in the past get less attention. Additionally, to ensure that the whole sentence is translated, it is required that the coverage value associated with all the source words become 1. The authors use this constraint as an auxiliary loss in the training objective.

Mi *et al.* (2016) also followed a similar strategy to increase coverage to mitigate repeating phrases. However, they associated a coverage embedding rather than a scalar with each source word to track if it has been entirely translated. Note that previous attention weights are incorporated in coverage vector and coverage embeddings through a GRU in both these models.

Sankaran *et al.* (2016) followed a simple approach to restrict the attention mechanism on again attending on the previously attended tokens. They aggregated the attention scores given to a source word. These aggregated scores modulated the score given to the corresponding source word at the current timestep.

For extractive document summarization, Chen *et al.* (2016) focused on diversifying the context vectors by subtracting the aggregated history of context vectors from the current context vector. This was primarily done to ensure that all the salient points in the document were being covered and no point was being repeated while identifying the relevant sentences.

See *et al.* (2017) observed repeating phrases in abstractive summarization. The authors adopted the technique from Tu *et al.* (2016) to ensure that the same source words are not attended to again. Note that instead of using a GRU to model the history of attention weights, the authors aggregate the attention distribution from the past time steps. Although they have used the auxiliary loss as given in Tu *et al.* (2016) to increase coverage, they acknowledge that full coverage is not needed for summarization. Therefore, they weigh the coverage loss with a parameter  $\lambda$  and use it to avoid repeated attention on the words which were attended to earlier.

Paulus *et al.* (2018) focused on abstractive summarization to tackle several issues prevalent in existing models: exposure bias, repeating phrases, and incoherent phrases. For the latter two, the authors utilize the method proposed in Sankaran *et al.* (2016) to

modulate the attention given to the source words based on the previous attention distributions. Additionally, they hypothesized that the repetitions could be due to similar decoder states in longer summaries. Therefore, the authors also introduced intra-decoder attention with temporal attention (Sankaran *et al.*, 2016). Along with these architectural changes, at inference time, they restricted the repetitions of tri-grams.

Note that in contrast to the above works, the primary focus of our proposed method in Chapter 4 is on avoiding repeating phrases rather than increasing coverage. For query-based abstractive summarization, it is not crucial to focus on all the parts of the input document. Instead, it is essential only to highlight the information relevant to the query. Therefore increasing coverage will hinder this end goal. Secondly, constraining the attention distribution to be different at each time step will not solve the purpose if the contextual representations associated with each source word are similar to each other. Therefore, we focus on diversifying the context vectors rather than the attention distributions.

*Miscellaneous Approaches:* Several other works (Gehrmann *et al.*, 2018; Chorowski and Jaitly, 2017; Suzuki and Nagata, 2017; Zhao *et al.*, 2018) also focus on the problem of avoiding repeating phrases. However, the strategy used in such works is different from tracking the attention on tokens. We give a brief overview of the strategies used in these works below.

Gehrmann *et al.* (2018) adopted a trick at the decoding stage, where they penalized the candidate generations if some source token is given a total attention score of more than 1, *i.e.*, coverage. Chorowski and Jaitly (2017) also adds the coverage term as a penalty in scoring the top-k outputs from beam search.

In contrast to making architectural changes in seq2seq models to avoid repeating phrases, Suzuki and Nagata (2017) avoids repeating phrases in abstractive summarization by predicting an upper bound on the frequency of each word in the target vocabulary while generating each summary. This upper bound is then taken into account during the decoding. Similarly, for question generation, Zhao *et al.* (2018), constrained the probability score coming from the copy network to a maximum of the number of occurrences of that word from the paragraph. The authors here hypothesized that the repetitions in the generated question are due to recurring words in the passage and thus

the solution.

## **Implicit Approaches**

Several works (Tu *et al.*, 2017a,b; Su *et al.*, 2018) started focusing on overall generation quality, which indirectly alleviated the problem of repeating phrases.

Tu *et al.* (2017a) observed that the generation of any word depends on the attended input representation (source context) and the previously generated words (decoder state or target context). The authors hypothesized that a generated sequence’s adequacy depends more on the source context, and fluency depends more on the previous words or target context. They observed that repeating phrases in machine translation occurred because the target context was not being taken into account adequately while generating the output. Thus, the authors propose a learnable gate that is modeled using a feed-forward neural network and depends on the source and target context. It dynamically weighs the contribution of target and source context in the final decoder state, which is then used to predict the next word. Note that this work assumes that source and target representations are appropriate, and the problem lies in efficiently fusing them. However, in this thesis, our focus is to improve upon the source context by diversifying it.

Tu *et al.* (2017b) added another module in the vanilla seq2seq model: encode-attend-decode-reconstruct, where the reconstruct model is responsible for predicting the source sentence from the generated output sequence. This constrained the model to translate all the words only once in the source sentence, to reconstruct it back, thus ensuring fewer repetitions and good coverage. The authors also used a context gate (Tu *et al.*, 2017a) and coverage vector (Tu *et al.*, 2016) alongside their model and saw that the performance improves. Note that it is difficult to adopt the reconstruct model for summarization, as it will be challenging to generate the whole document from a concise summary.

Su *et al.* (2018) proposed a Variational Recurrent Neural Networks based model for machine translation to improve the overall generation. The authors introduced a series of latent variables to capture complex dependencies among the translated words

at different timesteps and observed fewer repetitions in the translated sentences.

Recent works (Holtzman *et al.*, 2019; Jiang *et al.*, 2020) observed that this issue is still persistent in transformer-based seq2seq models and also for an open-ended generation. However, there has been a shift in the kind of approaches for transformer-based seq2seq models. The new strategies focus more on better auxiliary loss and decoding strategies than on architectural changes. One of the recent works concentrates on maximizing the negative samples' unlikelihood along with maximizing the likelihood for positive samples (Li *et al.*, 2019; Welleck *et al.*, 2019). Holtzman *et al.* (2019); Basu *et al.* (2021) ensure that the decoding strategies are adaptive and do not cause the generation to degenerate. Jiang *et al.* (2020) give a theoretical framework for understanding why repeating phrases is a prevalent problem. They suggest that the inherent repetition in any language is one reason for repetitions in the generated output.

### **3.2.2 Better Transition among fields for Structured data to Text Generation**

A smooth transition between the fields is necessary to generate coherent descriptions from structured data. Hence, various models were proposed (Sha *et al.*, 2017; Iso *et al.*, 2019; Puduppully *et al.*, 2019), similar to ours, (Nema *et al.*, 2018), that ensured a better transition between the fields. We describe some of them below.

Sha *et al.* (2017) focus on deciding the order of the fields that should be incorporated in the description. Along with the dual attention mechanism, they had a link-based attention mechanism that captured the interaction between the fields. These pairwise interactions acted as a transition matrix. These two kinds of attention were then combined using an adaptive gate. Kiddon *et al.* (2016) explored the use of checklists to track previously visited ingredients while generating recipes from ingredients.

Iso *et al.* (2019) keep track of fields that have already been mentioned. The authors focus on generating descriptions for NBA games from given game statistics. The average description length is 384 tokens. The model consists of two components: one for tracking saliency and the other for generating the text. The former is responsible for updating the state if the generation will transition to another entity, else the same state

is used for the entity that is being explained currently. As the output is lengthy, the same entity can be referred to multiple times. Therefore, they ensure that if the same entity is referred again, then the memory state of the last appearance is then passed to the decoder.

Puduppully *et al.* (2019) also focuses on generating game summaries from game statistics. The authors focus on content selection and content planning using two different modules. The content selection is modeled using intra-record level attention. It helps highlight corresponding fields that might be relevant while talking about an entity. For content planning, they first generate the order in which the entities will occur in a sequence. This ordered sequence is then passed to the text generation module, which renders the game description conditioned on this ordered sequence.

### 3.2.3 Improving answerability for Question Generation

Several works have observed that even if the answer is passed as an input, the generated question is (i) not specific to the given answer type (Sun *et al.*, 2018; Zhou *et al.*, 2019; Wang *et al.*, 2020a; Kim *et al.*, 2018) or (ii) copies irrelevant context from the source (Sun *et al.*, 2018), or (iii) copies the answer itself in the question (Kim *et al.*, 2018). This degrades the answerability of the question, and thus the following works have proposed different ways to mitigate this problem.

In addition to the question not being specific to the answer, Sun *et al.* (2018) observed that the question type did not match the answer type. To this end, the authors used a separate classifier to predict only the question type using the answer embedding. Also, the authors observed that irrelevant content words were often copied in the question. Therefore, they used a position-aware attention mechanism that constrained the model to copy words only from the surrounding context of the answer. Zhou *et al.* (2019) observed that even with passing answer as an input, the question type of the generation question still did not match the answer. To this end, the authors use a classifier to predict the question type using the answer representation. The classification loss was used as an auxiliary loss to train the network to predict the correct question type.

Kim *et al.* (2018) observed that answer keywords often get generated in the question.

Therefore, the authors mask the answer in the given passage and encode it separately. They further use the attention mechanism at both the passage and answer levels. The passage and answer-level context vectors are then used to generate the next word of the question. Wang *et al.* (2020a) also observed that the question type and answer type are different. To this end, they proposed a fusion gate to combine passage and answer representation, which in turn is used to initialize the decoder state. The authors also introduced a graph encoder based on GCNs, which encoded the dependency graph of the passage. The learnt graph-based representations are passed semantic rich fusion attention to encode the semantics of the passage. Similar to our work, they also used the REINFORCE algorithm with BLEU as a reward function to fine-tune the model. Ma *et al.* (2020a) believed that lack of global question semantics and inability to encode answer positions effectively leads to incoherent questions. This work also uses a fusion gate to initialize the decoder state. Moreover, they hypothesize that the generic questions are generated whenever the encoder reads the same sentence while generating two different questions. The authors mitigated this by introducing a sentence-level matching module that learns sentence-level semantics in both the encoder and the decoder.

### 3.2.4 Other Limitations

Apart from the above limitations, there are various other limitations of NLG models that have been explored in the existing literature. Below, we discuss some works which focus on these additional limitations observed in seq2seq models.

*Hallucinations* refer to the problem of generating words or phrases which are not supported by the source. This limitation has also been observed in several NLG tasks such as: structured data to text generation (Tian *et al.*, 2019; Nie *et al.*, 2019), abstractive summarization (Maynez *et al.*, 2020), Image Captioning (Rohrbach *et al.*, 2018), *etc.* Tian *et al.* (2019) hypothesize that the irrelevant content is generated when the input is not given enough attention. The authors propose a confidence score that ensures that the model assigns high attention weights to the input words when the content is copied. On the other hand, Nie *et al.* (2019) observed that hallucinations could occur if the model is trained on loosely tied (structured data, description) pairs. Therefore, they refine the training data using an NLU component and iterative self-training. Maynez *et al.* (2020)



assess hallucinations in abstractive summarization by conducting human evaluations at a large scale. They distinguish hallucinations in two categories: intrinsic, *i.e.*, manipulating information present in the input (e.g., “*a mayoral candidate*” in the source is referred to as “*mayor*” in target ), and extrinsic, *i.e.*, irrelevant context with respect to the source. The authors show that large pre-trained models like BERT based seq2seq models were more faithful and factually correct as compared to RNN based seq2seq models.

In automatic question generation, apart from generating questions specific to the answers, some works have focused on generating difficult questions. Gao *et al.* (2018) propose an AQG framework that generates a question corresponding to a given difficulty level (hard or easy). Similar to the above work, Kumar *et al.* (2019) also focuses on generating difficult questions. However, the input is a knowledge graph instead of text. Here, the authors estimate the difficulty level using named-entity popularity, which is used to generate a question. Ma *et al.* (2020b) focuses on generating complex multi-hop questions instead of generating simple questions. To this end, the authors construct an answer-centric graph from the given passage and then encode it using Graph Convolutional Networks (GCNs). Pan *et al.* (2020) focus on generating deep questions, such that answering such questions would require reasoning over multiple pieces of information. To this end, the authors create a semantic graph for the given passage, encode it using both GCNs and RNNs, and then select the appropriate context to generate the question. The authors conduct a human study to validate that the generated questions were indeed complex.

For abstractive summarization, it is important that *all* the salient information present in the document should be summarized. Several works (Gehrmann *et al.*, 2018; Chen and Bansal, 2018; Saito *et al.*, 2020) point out that although models generate fluent summaries, they often do not select all the salient sentences before generating the summaries. Thus, the above works add an intermediate step of selecting the sentences and tokens that should be summarized and then pass the selected sentences to the decoder to generate an abstractive summary. Gehrmann *et al.* (2018) use a supervised approach to classify whether a token is salient. Chen and Bansal (2018) select the salient information at sentence-level using policy gradient algorithm, with ROUGE score as the reward

signal. Saito *et al.* (2020) combine the extraction approach from Gehrmann *et al.* (2018) with a large pre-trained model (Lewis *et al.*, 2019) to select salient sentences.

### **3.3 Works using similar techniques as proposed in this thesis**

To mitigate the three main limitations discussed above, we propose techniques that use (i) orthogonalization to diversify context vectors, (ii) explicit reward signals to improve generation, and (iii) multiple decoders to generate refined drafts. These techniques are generic and have been used for other purposes also as discussed below.

#### **3.3.1 Using Orthogonalization to Diversify Context Vectors**

To avoid repeating phrases and simulate the *never look back* aspect in generating descriptions from structured data, we resort to diversifying context vectors by orthogonalizing the current context vector to the history of context vectors. This strategy was later adopted in other NLG tasks.

For Question Answering based on Reading Comprehension, Parikh *et al.* (2018) adopt a similar strategy for answering multiple-choice questions. The authors follow the elimination strategy to arrive at the correct option. To this end, the authors iteratively eliminate the options by subtracting the eliminated option’s projection from the passage representation. They observed that this strategy was effective for RACE (Lai *et al.*, 2017) compared to simply predicting the correct option in one pass.

For various NLP tasks, there has been an ongoing debate (Bastings and Filippova, 2020) on whether the attention distributions learnt are faithful to the model’s predictions. Mohankumar *et al.* (2020) identified that one of the reasons for the attention distributions not being faithful is that the underlying contextual representations of different input tokens are very similar. To this end, one of the solutions proposed was to orthogonalize the next token’s representation to the representations of the previous tokens. This helped in making the attention mechanism more transparent and interpretable.

### 3.3.2 Using explicit reward signals for improving NLG

Seq2seq models are typically trained by maximizing the log-likelihood. At training time, the model predicts the next word conditioned on the words given in the corresponding ground truth sequence. On the other hand, at test time, the next word is based on the words previously predicted by the model. This discrepancy in decoding at training and testing time leads to the exposure bias problem (Ranzato *et al.*, 2016).

Several works (Ranzato *et al.*, 2016; Rennie *et al.*, 2017; Paulus *et al.*, 2018; Song *et al.*, 2017; Pasunuru and Bansal, 2018; Yuan *et al.*, 2017) approached this problem by training the models with a combined objective function: a weighted objective of maximizing the log likelihood and rewarding the decoder using external reward signals through REINFORCE with baseline algorithm (Williams, 1992). The typical rewards used in these works (Ranzato *et al.*, 2016; Rennie *et al.*, 2017; Pasunuru and Bansal, 2018; Paulus *et al.*, 2018; Song *et al.*, 2017) are  $n$ -gram based metrics (BLEU, ROUGE, CIDEr). These works cover a wide variety of tasks such as image captioning (Rennie *et al.*, 2017), abstractive summarization (Paulus *et al.*, 2018), question generation (Song *et al.*, 2017; Yuan *et al.*, 2017), etc.

However, only a few works have utilized this approach to improve upon task-specific aspects. For example, Song *et al.* (2017) use BLEU as a reward for question generation, although it only improves the fluency of a question, and not other aspects such as answerability, difficulty level, *etc.* On the other hand, Yuan *et al.* (2017) design the reward as a combination of the QA performance on the generated question and perplexity of the generated question. One limitation of this work is that the QA model’s performance may not always be the right indicator of the quality of the question. For example, Jia and Liang (2017); Weissenborn *et al.* (2017) point out that deep learning based QA systems achieve high performance by identifying the right keywords in the question. However, the question could still be incoherent.

Parallel to our work, Pasunuru and Bansal (2018) use task-specific aspects in the reward function for abstractive summarization. The authors propose some modifications in the ROUGE metric to better reward the model for saliency by weighing some keyphrases more than the others. Second, they use a state-of-the-art entailment model

to quantify the entailment of the generated summary from the ground summary. These two rewards are then utilized with REINFORCE algorithm to train the model.

### 3.3.3 Generating multiple drafts

Seq2seq models typically predict the target sequence in a single pass. Consequently, small mistakes at the start of the sequence can adversely impact the rest of the generated output. Xia *et al.* (2017) was one of the initial works that proposed generating the target sequence in multiple passes for machine translation. The authors generated the target sequence in two passes. They use Monte Carlo based training method to train the network. In contrast, we maximize the log-likelihood of both the initial and final draft independently.

Wang *et al.* (2018a) proposed a slightly different approach to write and edit the generated draft. This approach was mainly proposed for writing abstracts from a title. The first pass was generated using a standard decoder. The generated abstract was then passed through a bidirectional LSTM network, and a learnable gate was used to adaptively attend to the generated abstract and the title. This new attended representation, which is a fusion of the initial draft of the abstract and the title, is again passed to the decoder to generate the second draft.

Zhang *et al.* (2018) emphasized that current seq2seq models only take left-to-right context while generating a sentence. To also account for the right-to-left context, the authors generate the first draft from right to left. This reverse draft and the source input are then taken into account while generating the second draft from left to right. Similarly, Geng *et al.* (2018) point out that one pass decoding does not take the global information into account while generating the sequence in the case of Machine Translation. To this end, the authors propose a multi-pass decoder that keeps refining the draft based on source and the last generated output until the BLEU score keeps increases. The decision to whether generate a new draft or stop refining is modeled using a policy network. The reward for the same is the BLEU score between the final generated output and the ground truth.

## 3.4 Summary

This chapter gave a brief overview of other works related to the tasks addressed in this thesis, the limitations addressed in this thesis, and the techniques used in this thesis. In particular, we discussed several models proposed in the pre-deep learning and deep learning eras for different tasks addressed in this work, *viz.*, structured data to text generation, question generation, and query-based abstractive summarization. We then gave a quick primer on relevant works that also mitigate the limitations considered in this work, *viz.*, avoiding repeating phrases, better transition among fields in structured data to text generation, and improving answerability in question generation. Finally, we discussed works that also use similar techniques as those proposed in our works, *viz.*, orthogonalization for diversifying context vectors, explicit reward signals for training seq2seq models, and generation in multiple drafts.

## CHAPTER 4

### Avoiding Repeating Phrases in NLG

In this chapter, we describe how the REFINE module is designed to avoid the problem of repeating phrases in the context of Query Based Abstractive Summarization.

#### 4.1 Introduction

As mentioned earlier, seq2seq models have led to significant improvements across a wide variety of NLG tasks. However, these models are still limited and generate incoherent sentences (as illustrated in Chapter 1). One of the recurring problems is that the generated outputs contain repeating phrases. Such repetitions are not specific to any particular NLG task. Instead, they have been observed across a wide variety of tasks, such as Machine Translation (Sankaran *et al.*, 2016), Abstractive Summarization (See *et al.*, 2017), and open-ended generation (Holtzman *et al.*, 2019).

When we, as humans, generated a description or a summary, we keep track of the information that we have already conveyed and avoid repeating it. Such a module that explicitly prevents the information from being repeated is missing in existing encode-attend-decode models. More specifically, a typical encode-attend-decode model first computes a vectorial representation for the passage and the query and then produces a contextual summary one word at a time. Each word is produced by feeding a new context vector to the decoder at each time step by attending to different parts of the passage and query. If the decoder produces the same word or phrase repeatedly, then it could mean that the context vectors fed to the decoder at these time steps are very similar. A similar hypothesis is also proposed in Sankaran *et al.* (2016) where they say that the repeating phrases could be due to the decoder state being similar at different timesteps while generating lengthy summaries.

To alleviate this problem, we propose a model which explicitly prevents this by ensuring that successive context vectors are orthogonal to each other. Specifically, we

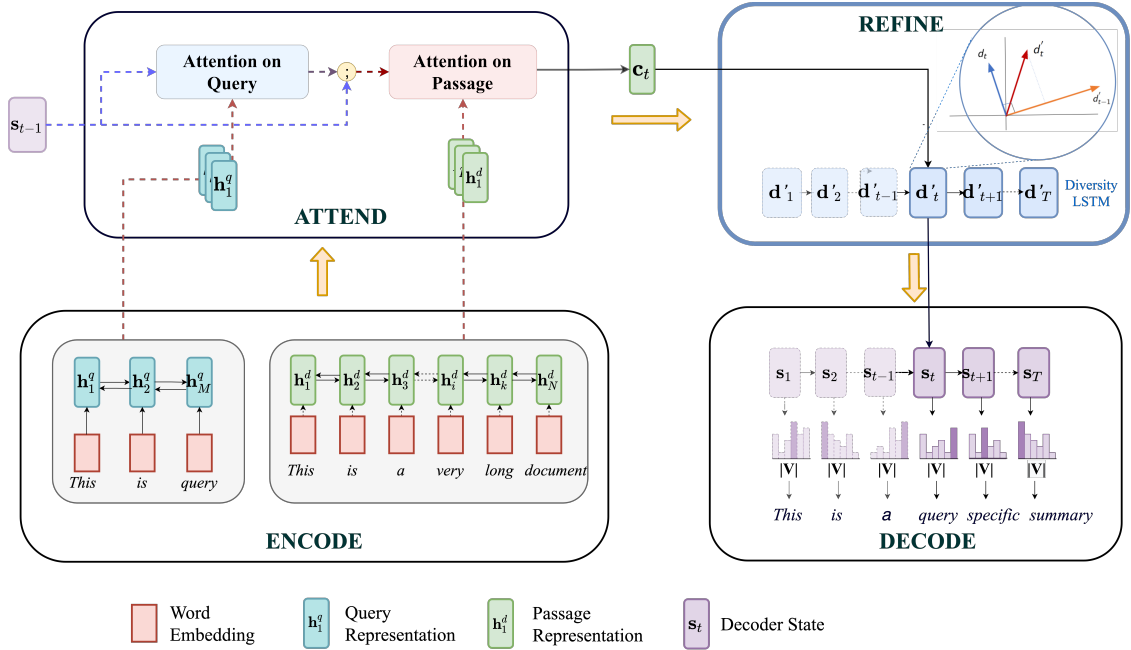


Figure 4.1: Our proposed model is based on encode-attend-refine-decode paradigm. The attend module generates a context vector  $c_t$ , by attending on to the passage word representations based on the decoder’s hidden state  $s_{t-1}$ . This context vector  $c_t$ , is then passed to the refine module, which diversifies  $c_t$  with respect to the history of the context vectors  $d'_{t-1}$  stored in the refine module. The diversified context vector  $d_t$  is then passed to the decode module to generate the next word.

subtract out any component that the current context vector has in the direction of the previous context vector. Notice that we do not require the current context vector to be orthogonal to all previous context vectors but just its immediate predecessor. This enables the model to attend to words repeatedly if needed later in the process. To account for the complete history (or all previous context vectors) we also propose an extension of this idea where we pass the sequence of context vectors through a LSTM network (Hochreiter and Schmidhuber, 1997) and ensure that the current state produced by the LSTM is orthogonal to the history. We refer to this as the Diversity LSTM cell. At each time step, the state of the Diversity LSTM cell is then fed to the decoder to produce one word in the summary.

Note that in some situations, completely orthogonalizing the current context vector to the previous context vector(s) may not be ideal. For example, in machine translation, one word in the source could be aligned to multiple words in the target. In such cases, it is crucial that we attend to the corresponding source word and do not diversify until the whole phrase in the target has been generated. Hence, it is essential to dynamically

decide the amount of previous context vector that should be subtracted from the current attended context vector. For this, we introduce a learnable gate that allows for soft orthogonalization, *i.e.*, it ensures that the new context vector is not completely orthogonal to the previous context vector(s). Figure 4.1 shows the complete architecture of our model containing the *refine* module.

We evaluate the proposed refinements in the context of query-based abstractive summarization. This task was not much explored when we started this work. As opposed to abstractive summarization, where the goal is to cover all the salient points in a passage, in query based summarization, the focus is on generating a summary that is relevant to the query. Thus given a passage on “*the super bowl*”, the query “*How was the half-time show?*” would result in a summary that would not cover the actual game itself. Note that there had been some work on query-based extractive summarization (Cao *et al.*, 2016; Ma *et al.*, 2016; Liu *et al.*, 2012) earlier where the aim was to extract the most salient sentence(s) from a passage specific to the given query and treat these as a summary. There is no natural language generation involved. Since we were interested in abstractive (as opposed to extractive) summarization, we created a new dataset based on Debatepedia. This dataset contains triplets of the form (query, passage, summary). Further, each summary is abstractive and not extractive, *i.e.*, the summary does not necessarily comprise of a sentence/phrase simply copied from the original passage. Note that, for this task as well, we observed the problem of repeating phrases in the outputs of existing NLG systems (as illustrated in Table 4.1).

Our contributions, as discussed in this chapter, can be summarized as follows: (i) We propose a new model based on encode-attend-refine-decode paradigm to avoid repeating phrases; (ii) We propose a new dataset for query-based abstractive summarization; (iii) We study the problem of repeating phrases in NLG in the context of this dataset. We show that our method outperforms a vanilla sequence-to-sequence model with a gain of 28% (absolute) in ROUGE-L score;(iv) We also demonstrate that our method clearly outperforms a parallel method proposed for handling the problem of repeating phrases with a gain of 7% (absolute) in ROUGE-L scores (v) We do a qualitative analysis of the results and demonstrate that our model indeed produces outputs with fewer repetitions.



<p><b>Passage Snippet:</b> The “natural death” alternative to euthanasia is not keeping someone alive via life support until they die on life support. That would, indeed, be unnatural. The natural alternative is, instead, to allow them to die off of life support.</p> <p><b>Query:</b> Is euthanasia better than withdrawing life support (non-treatment)?</p> <p><b>Ground Truth Summary:</b> The alternative to euthanasia is a natural death without life support.</p> <p><b>Predicted Summary:</b> the large to euthanasia is a natural death <b>life life</b> use</p>
<p><b>Passage Snippet:</b> Legalizing same-sex marriage would also be a recognition of basic American principles and represent the culmination of our nation’s commitment to equal rights. Some have said, the last major civil-rights milestone yet to be surpassed in our two-century struggle to attain the goals we set for this nation at its formation.</p> <p><b>Query:</b> Is gay marriage a civil right?</p> <p><b>Ground Truth Summary:</b> Gay marriage is a fundamental equal right.</p> <p><b>Predicted Summary:</b> gay marriage is a appropriate <b>right right</b></p>

Table 4.1: Examples showing repeated words in the output of encoder-decoder models.

## 4.2 Model Architecture

This section describes the design for all the modules: encode, attend, refine, and decode used in our proposed model to avoid repeating phrases for Query Based Abstractive Summarization.

Given a passage  $\mathbf{P} = \{w_1^p, \dots, w_n^p\}$  containing  $n$  words, and a query  $\mathbf{Q} = \{w_1^q, \dots, w_k^q\}$  containing  $k$  words, the task of generating a summary  $\mathbf{y} = \{y_1, y_2, \dots, y_m\}$  for the given (passage,query) pair can be modeled as the problem of finding a  $\mathbf{y}^*$  that maximizes the probability  $p(\mathbf{y}|\mathbf{Q}, \mathbf{P})$  which can be further decomposed as:

$$\mathbf{y}^* = \arg \max_{\mathbf{y}} \prod_{t=1}^m p(y_t | y_1, \dots, y_{t-1}, \mathbf{P}, \mathbf{Q}) \quad (4.1)$$

We now describe a way of modeling  $p(y_t | y_1, \dots, y_{t-1}, \mathbf{P}, \mathbf{Q})$  using the proposed *encode-attend-refine-decode* paradigm. The proposed model contains the following components: (i) an encoder RNN for the query and the passage (ii) attention mechanism for the query and the passage and (iii) a refine module to modify the contextual representations and (iv) a decoder RNN to generate the output, one word at a time. All the RNNs use a GRU cell.

### 4.2.1 Encode

The encode module is responsible for learning contextual representation for both passage and query. Our encode module consists of two layers:

*Embedding Layer:* Each word in the passage and the query is embedded in a  $d$ -dimensional space using word-level embeddings.

*Contextual Layer:* To encode the contextual information for each word in the passage and query, we pass the word embeddings through a bidirectional LSTM network. This ensures that the context before and after the word is encoded in the learnt representation.

$$\vec{\mathbf{h}}_i^p = \text{LSTM}(\mathbf{e}(w_i^p), \vec{\mathbf{h}}_{i-1}^p) \quad (4.2)$$

$$\overleftarrow{\mathbf{h}}_i^p = \text{LSTM}(\mathbf{e}(w_i^p), \overleftarrow{\mathbf{h}}_{i-1}^p) \quad (4.3)$$

$$\vec{\mathbf{h}}_i^q = \text{LSTM}(\mathbf{e}(w_i^q), \vec{\mathbf{h}}_{i-1}^q) \quad (4.4)$$

$$\overleftarrow{\mathbf{h}}_i^q = \text{LSTM}(\mathbf{e}(w_i^q), \overleftarrow{\mathbf{h}}_{i-1}^q) \quad (4.5)$$

where  $\mathbf{e}(\cdot) \in \mathbb{R}^d$  is the  $d$ -dimensional word embedding for the given word. The passage contextual representations are denoted using  $\mathbf{H}^p = \{\mathbf{h}_1^p, \dots, \mathbf{h}_n^p\}$ , and query  $\mathbf{H}^q = \{\mathbf{h}_1^q, \dots, \mathbf{h}_k^q\}$

### 4.2.2 Attend

We know that attend module helps the decode module generate the next word by selecting the *next* relevant context from the passage. However, for query-based abstractive summarization, the relevant context must be specific to the given query. To account for this, we propose a dual attention mechanism.

**Attention mechanism for the query:** At each time step, the decoder produces an output word by focusing on different portions of the query (passage) with a query (passage) attention model. We first describe the query attention model, which assigns

weights  $\alpha_{t,i}^q$  to each word in the query at each decoder time step  $t$  as follows:

$$a_{t,i}^q = \mathbf{v}_q^T \cdot \tanh(\mathbf{W}_q \cdot \mathbf{s}_t + \mathbf{U}_q \cdot \mathbf{h}_i^q) \quad (4.6)$$

$$\alpha_{t,i}^q = \frac{\exp(a_{t,i}^q)}{\sum_{j=1}^k \exp(a_{t,j}^q)} \quad (4.7)$$

where  $\mathbf{s}_t$  is the current state of the decoder at time step  $t$  (we will see an exact formula for this soon).  $\mathbf{W}_q \in \mathbb{R}^{l_2 \times l_1}$ ,  $\mathbf{U}_q \in \mathbb{R}^{l_2 \times l_2}$ ,  $\mathbf{v}_q \in \mathbb{R}^{l_2}$  are parameters.  $l_1$  is the size of the decoder's hidden state,  $l_2$  is both the size of  $\mathbf{h}_i^q$  and also the size of the final query representation at time step  $t$ , which is computed as:

$$\mathbf{q}_t = \sum_{i=1}^k \alpha_{t,i}^q \mathbf{h}_i^q \quad (4.8)$$

**Attention mechanism for the passage :** We now describe the passage attention model, which assigns weights to each word in the passage using the following attention model.

$$a_{t,i}^p = \mathbf{v}_d^T \cdot \tanh(\mathbf{W}_d \cdot \mathbf{s}_t + \mathbf{U}_d \cdot \mathbf{h}_i^p + \mathbf{Z}_d \cdot \mathbf{q}_t) \quad (4.9)$$

$$\alpha_{t,i}^p = \frac{\exp(a_{t,i}^p)}{\sum_{j=1}^n \exp(a_{t,j}^p)}$$

where  $\mathbf{s}_t$  is the current state of the decoder at time step  $t$  (we will introduce an exact formula for this soon).  $\mathbf{W}_d \in \mathbb{R}^{l_4 \times l_1}$ ,  $\mathbf{U}_d \in \mathbb{R}^{l_4 \times l_4}$ ,  $\mathbf{Z}_d \in \mathbb{R}^{l_4 \times l_2}$ ,  $\mathbf{v}_d \in \mathbb{R}^{l_4}$ ,  $l_4$  is the size of  $\mathbf{h}_i^p$  and also the size of the final passage representation  $\mathbf{c}_t$  which is passed to the decoder at time step  $t$  as:

$$\mathbf{c}_t = \sum_{i=1}^n \alpha_{t,i}^p \mathbf{h}_i^p \quad (4.10)$$

Note that  $\mathbf{c}_t$  now encodes the relevant information from the passage as well as the query (refer Equation (4.9)) at time step  $t$ . We refer to this as the *context vector* for the decoder.

### 4.2.3 Refine

As hypothesized earlier, if the decoder produces the same phrase/word multiple times, then the context vectors being fed to the decoder at consecutive time steps may be very similar. The goal of the refine module is to diversify the context vector based on the prior context vector. It is designed as follows:

$$\mathbf{d}_t = \text{refine}(\mathbf{c}_t, \mathbf{d}_{t-1}) \quad (4.11)$$

where  $\mathbf{c}_t$  is the passage context vector generated from the attend module.  $\mathbf{d}_t$  is the output from the refine module and is the final context vector seen by the decoder at timestep  $t$ . We propose four variants of the refine module as discussed below.

- **D<sub>1</sub>**: In this variant, we simply make context vector  $\mathbf{c}_t$  orthogonal *only* to its immediate predecessor, *i.e.*  $\mathbf{d}_{t-1}$  :

$$\begin{aligned} \mathbf{d}_t &= \text{refine}(\mathbf{c}_t, \mathbf{d}_{t-1}) \\ &= \mathbf{c}_t - \frac{\mathbf{c}_t^T \mathbf{d}_{t-1}}{\mathbf{d}_{t-1}^T \mathbf{d}_{t-1}} \mathbf{d}_{t-1} \end{aligned} \quad (4.12)$$

- **SD<sub>1</sub>**: The above variant imposes a hard orthogonality constraint on the context vector, that could potentially affect the generation quality in some situations. We therefore propose a relaxed version of the above variant which uses a learnable gate. This gate can dynamically decide what fraction of the previous context vector should be subtracted from the current context vector using the following equations:

$$\gamma_t = \mathbf{W}_g \cdot \mathbf{d}_{t-1} + \mathbf{b}_g \quad (4.13)$$

$$\mathbf{d}_t = \text{refine}(\mathbf{c}_t, \mathbf{d}_{t-1}) \equiv \mathbf{c}_t - \gamma_t \frac{\mathbf{c}_t^T \mathbf{d}_{t-1}}{\mathbf{d}_{t-1}^T \mathbf{d}_{t-1}} \mathbf{d}_{t-1} \quad (4.14)$$

where  $\mathbf{W}_g \in \mathbb{R}^{l_4 \times l_4}$ ,  $\mathbf{b}_g \in \mathbb{R}^{l_4}$  are learnable parameters and  $l_4$  is the dimension of  $\mathbf{d}_t$ .

- **D<sub>2</sub>**: In our preliminary analysis, we observed that in some cases a phrase can repeat after some interval, *i.e.*, after a few words have been generated in between.

Therefore, it is important to ensure that the current context vector is also diverse to *all* the previous context vectors. The above two proposed refinements do not take the history into account. To account for the history, we pass the successive context vectors as a sequence through a novel variant of LSTM. We refer to this proposed variant as the Diversity LSTM cell. Along with learning a meaningful representation of the context vector sequence, it ensures that the new state of the Diversity LSTM cell is orthogonal to the maintained cell state. Specifically, we use the following set of equations to compute a diverse context vector at time  $t$ :

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot \mathbf{c}_t + \mathbf{U}_i \cdot \mathbf{d}'_{t-1} + \mathbf{b}_i) \quad (4.15)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot \mathbf{c}_t + \mathbf{U}_f \cdot \mathbf{d}'_{t-1} + \mathbf{b}_f) \quad (4.16)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot \mathbf{c}_t + \mathbf{U}_o \cdot \mathbf{d}'_{t-1} + \mathbf{b}_o) \quad (4.17)$$

$$\hat{\mathbf{d}}_t = \tanh(\mathbf{W}_d \cdot \mathbf{c}_t + \mathbf{U}_d \cdot \mathbf{d}'_{t-1} + \mathbf{b}_d) \quad (4.18)$$

$$\mathbf{d}'_t = \mathbf{i}_t \odot \hat{\mathbf{d}}_t + \mathbf{f}_t \odot \mathbf{d}'_{t-1} \quad (4.19)$$

$$\mathbf{d}_t^{diverse} = \mathbf{d}'_t - \frac{\mathbf{d}'_t{}^T \mathbf{d}'_{t-1}}{\mathbf{d}'_{t-1}{}^T \mathbf{d}'_{t-1}} \mathbf{d}'_{t-1} \quad (4.20)$$

$$\mathbf{d}_t = \mathbf{o}_t \odot \tanh(\mathbf{d}_t^{diverse}) \quad (4.21)$$

where  $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_c \in \mathbb{R}^{l_5 \times l_4}$ ,  $\mathbf{U}_i$  and  $\mathbf{U}_f, \mathbf{U}_o, \mathbf{U}_d \in \mathbb{R}^{l_5 \times l_4}$  are learnable parameters. The vector  $\mathbf{c}_t$  is the  $l_4$ -dimensional output of Equation (4.10) and  $l_5$  is the number of hidden units in the Diversity LSTM cell. This final  $\mathbf{d}_t$  from Equation (4.21) is then passed on to the decoder. Note that Equation (4.20) ensures that the state of the LSTM at time step  $t$  is orthogonal to the previous history, *i.e.*,  $\mathbf{d}'_{t-1}$ . Figure 4.1 shows a pictorial representation of the model with a diversity LSTM cell.

- **SD<sub>2</sub>**: Here, we again introduce a learnable gate which controls the fraction of the component along  $\mathbf{d}'_{t-1}$  that should be subtracted from  $\mathbf{d}'_t$ . Specifically, we define a gating parameter  $g_t$  and replace (4.20) above by (4.23) as define below:

$$\gamma_t = \sigma(\mathbf{W}_g \cdot \mathbf{c}_t + \mathbf{U}_g \cdot \mathbf{d}'_{t-1} + \mathbf{b}_o) \quad (4.22)$$

$$\mathbf{d}_t^{diverse} = \mathbf{d}'_t - \gamma_t \frac{\mathbf{d}'_t{}^T \mathbf{d}'_{t-1}}{\mathbf{d}'_{t-1}{}^T \mathbf{d}'_{t-1}} \mathbf{d}'_{t-1} \quad (4.23)$$

where  $\mathbf{W}_g \in \mathbb{R}^{l_5 \times l_4}$ ,  $\mathbf{U}_g \in \mathbb{R}^{l_5 \times l_4}$  are learnable parameters.

#### 4.2.4 Decode

The decoder generates the output sequence one word at a time. The hidden state of the decoder  $\mathbf{s}_t$  at each time  $t$  is again computed using a GRU as follows:

$$\mathbf{s}_t = \text{GRU}_{dec}(\mathbf{s}_{t-1}, [\mathbf{e}(y_{t-1}), \mathbf{d}_{t-1}]) \quad (4.24)$$

where,  $\mathbf{e}(y_{t-1})$  is the  $d$ -dimensional of the word  $y_t$ , which has the highest probability under the following distribution over the vocabulary words at timestep  $t - 1$  computed as:

$$\tilde{p}(y_t) = \text{softmax}(\mathbf{W}_o \cdot f(\mathbf{W}_{dec} \cdot \mathbf{s}_t + \mathbf{V}_{dec} \cdot \mathbf{d}_t)) \quad (4.25)$$

where  $\mathbf{W}_o \in \mathbb{R}^{N \times l_1}$ ,  $\mathbf{W}_{dec} \in \mathbb{R}^{l_1 \times l_1}$ ,  $\mathbf{V}_{dec} \in \mathbb{R}^{l_1 \times l_4}$ ,  $V$  is the vocabulary size. The above output is exactly the quantity defined in Equation (4.1) that we wanted to model ( $p(y_t|y_1, \dots, y_{t-1}, \mathbf{P}, \mathbf{Q})$ ). Also  $[\mathbf{e}(y_{t-1}), \mathbf{d}_{t-1}]$  means a concatenation of the vectors  $\mathbf{e}(y_{t-1})$ ,  $\mathbf{d}_{t-1}$ . We chose  $f$  to be the identity function.

### 4.3 Baseline Methods

We compared our proposed model with a parallel work by Chen *et al.* (2016). This work focused on the task of extractive document summarization. They used the encode-attend-decode paradigm, where the authors *distracted* the context vector to ensure that the whole passage is taken into account for a better summary. Although the motivation is slightly different as they focus on coverage and not repetition, the method to distract the context vector is similar. The encode, attend and decode module is similar to the one described in this chapter. The authors model the *distraction* at two levels: (i) they ensure that the new context vector is different from the ones generated previously. In contrast to our method of orthogonalization, they use subtraction to ensure each context vector is different. (ii) they ensure that the attention paid to a token  $t$  is less if it has

been attended to in the past. We describe the two variants proposed by them below:

**M1:** This model accumulates all the previous context vectors as  $\sum_{j=1}^{t-1} \mathbf{d}_j$  and incorporates this history while computing a diverse context vector:

$$\mathbf{d}_t = \tanh(\mathbf{W}_c \cdot \mathbf{c}_t - \mathbf{U}_c \cdot \sum_{j=1}^{t-1} \mathbf{d}_j) \quad (4.26)$$

where  $\mathbf{W}_c, \mathbf{U}_c \in \mathbb{R}^{l_4 \times l_4}$  are diagonal matrices. We then use this diversity driven context  $\mathbf{d}_t$  in Equation (4.24) and (4.25).

**M2:** In this model, in addition to computing a diverse context as described in Equation (4.26), the attention weights at each time step are also forced to be diverse from the attention weights at the previous time step.

$$\alpha'_{t,i} = \mathbf{v}_a^T \cdot \tanh(\mathbf{W}_a \cdot \mathbf{s}_t + \mathbf{U}_a \cdot \mathbf{c}_t - \mathbf{b}_a \cdot \sum_{j=1}^{t-1} \alpha'_{j,i}) \quad (4.27)$$

where  $\mathbf{W}_a \in \mathbb{R}^{l_1 \times l_1}$ ,  $\mathbf{U}_a \in \mathbb{R}^{l_1 \times l_4}$ ,  $\mathbf{b}_a, \mathbf{v}_a \in \mathbb{R}^{l_1}$  are learnable parameters and  $l_1$  is the number of hidden units in the decoder GRU. Once again, they maintain a history of attention weights and compute a diverse attention vector by subtracting the history from the current attention vector. To have a fair comparison with the above two approaches, we adopt the above design in our refine module, keeping the other modules the same as described earlier.

## 4.4 Query Based Abstractive Summarization Dataset

As mentioned earlier, we primarily focused on the task of Query Based Abstractive Summarization. However, at the time of this work, there were no datasets that are specifically designed for query-based abstractive summarization. Therefore, we first create one such dataset from Debatepedia. Debatepedia is an encyclopedia of pro and con arguments and quotes on critical debate topics. There are 663 debates in the corpus (we considered only those debates which have at least one query with one passage). These 663 debates belong to 53 overlapping categories: *{Politics, Law, Crime, Environment, Health, Morality, etc.}* A given topic can belong to more than one category. For

Emissions: Is algae biofuel good for combating global warming?

---

Economics: Is algae biofuel economically viable?

---

Land-use: Does algae biofuel take up too much land?

---

Ecosystems: Is algae biofuel generally good for ecosystems?

---

Water-use: Does algae biofuel use too much water?

---

Clean coal: Is the use of algae to clean coal a good idea?

---

Vs. solar: Is algae biofuel superior to solar power?

---

Vs. other biofuels: Is algae biofuel superior to other biofuels?

---

Figure 4.2: Queries associated with the topic “algae biofuel”.

### Land-use: Does algae biofuel take up too much land?

- **Algae yields much more biofuel per acre than other fuels** Compared with second generation biofuels, algae are high-yield high-cost (30 times more energy per acre than terrestrial crops) feedstocks to produce biofuels. Since the whole organism uses sunlight to produce lipids, or oil, algae can produce more oil in an area the size of a two-car garage than an entire football field of soybeans.
- **Algae photo-bioreactors require very little land** "Algae: Not Only The Best Biofuel By Far...". [Ecoversity](#) - "For the algae-culture projects which use large growing ponds, the potential biodiesel production per acre is 30 to 100 times greater than obtainable with corn, soy and palm oil. However the most efficient systems, called photo-bioreactors, stack clear tubes of water with algae in the sun, requiring very little acreage for significant production. This is the system we are demonstrating at Ecoversity."

Figure 4.3: Passages and summaries for a given query.



<b>Average number of words per</b>		
Passage	Summary	Query
66.4	11.16	9.97

Table 4.2: Average length of passages/queries/summaries in the dataset.

example, the topic “*Eye for an Eye philosophy*” belongs to both “*Law*” and “*Morality*”. The average number of queries per debate is five, and the average number of passages per query is four.

For example, Figure 4.2 shows the queries associated with the topic “*Algae Biofuel*”. Figure 4.3 shows one sample query and the abstractive summary for some passages related to this query. Each passage is preceded by a crisp summary for the argument presented in it. We adopt the same as the summary for the passage. As is evident from the example, the summary is abstractive in nature and not extracted directly from the passage. We crawled 12,695 such {query, passage, summary} triples from Debatepedia (these were all the available triples). Table 4.2 reports the average length of the query, summary, and passages in this dataset.

We used ten-fold cross-validation for all our experiments. Each fold uses 80% of the passages for training, 10% for validation, and 10% for testing.

## 4.5 Experimental Setup

We evaluate our models on the dataset described in Section 4.4. Note that there were no prior baselines on query-based abstractive summarization, so we could only compare the models described above with different variants. Specifically, we compare the performance of the following models:

- **EAD:** This is the vanilla encode-attend-decode paradigm adapted for the task of abstractive query summarization. The encode and decode module has the same design as mentioned in Section 4.2.1 and Section 4.2.4 respectively. The attend module is minimal and only contains “**Attention Mechanism for the passage**” without taking the query into account in Equation 4.9. Along with helping us

understand the importance of the refine module, evaluating this model would also help us understand the importance of using a separate query attention mechanism.

- **EAD-Query**: This model is similar to the one described above, with the only addition of adding  $\mathbf{q}_t$  in Equation 4.9. Note that it still does not contain an “**Attention Mechanism for the query**”.
- **EAD-Query<sub>att</sub>**: This model contains the encode, attend and decode modules as described in Section 4.2, but it does not contain the refine module. In particular, instead of passing  $\mathbf{d}_{t-1}, \mathbf{d}_t$  to the decoder in Equation (4.24) and Equation (4.25) respectively, we simply pass the unrefined  $\mathbf{c}_{t-1}$  and  $\mathbf{c}_t$  as computed in Equation (4.10) to the decoder in Equations (4.24) and (4.25).
- **EARD-D<sub>1</sub>**: This is our proposed model, with the refine module designed based on Equation 4.12, where the context vector is diversified only based on the immediate predecessor and is strictly orthogonal to it. Note that the other modules encode, attend, decode are the same as in **EAD-Query<sub>att</sub>**.
- **EARD-D<sub>2</sub>**: This is another variant of our proposed model. The refine module here is Diversity LSTM cell with hard orthogonality, as described in Equation 4.15.
- **EARD-SD<sub>1</sub>**: This is similar to **EARD-D<sub>1</sub>** with a learnable gate for soft orthogonalization as mentioned in 4.14.
- **EARD-SD<sub>2</sub>**: This is similar to **EARD-D<sub>2</sub>** with a learnable gate for soft orthogonalization as mentioned in 4.23.
- **EARD-B<sub>1</sub>**: This baseline is based on the encode-attend-refine-decode paradigm. However, here we replace Diversity LSTM cell with a basic LSTM cell (i.e.,  $\mathbf{d}_t^{diverse} = \mathbf{d}'_t$  instead of using Equation (4.20)). This helps us understand whether simply using an LSTM to track the history of context vectors (without imposing a diversity constraint) is sufficient or not.
- **EARD-M<sub>1</sub>**: This baseline model is again based on the encode-attend-refine-decode paradigm. We adopt the distraction based design to diversify the context vector (as mentioned in Section 4.3).

S.No.	Models	ROUGE-1	ROUGE-2	ROUGE-L
encode-attend-decode based models				
1.	<b>EAD</b>	13.73	2.06	12.84
2.	<b>EAD-Query</b>	20.87	3.39	19.38
3.	<b>EAD-Query<sub>att</sub></b>	29.28	10.24	28.21
Baselines with encode-attend-refine-decode paradigm				
4.	<b>EARD-B<sub>1</sub></b>	23.18	6.46	22.03
5.	<b>EARD-M<sub>1</sub></b>	33.06	13.35	32.17
6.	<b>EARD-M<sub>2</sub></b>	18.42	4.47	17.45
Ours				
7.	<b>EARD-D<sub>1</sub></b>	33.85	13.65	32.99
8.	<b>EARD-SD<sub>1</sub></b>	31.36	11.23	30.5
9.	<b>EARD-D<sub>2</sub></b>	38.12	16.76	37.31
10.	<b>EARD-SD<sub>2</sub></b>	41.26	18.75	40.43
11.	+ Beam Search (k=5)	<b>42.12</b>	<b>19.01</b>	<b>40.83</b>
12.	<b>SOTA Su et al. (2021)</b>	<b>59.02</b>	<b>44.59</b>	<b>57.44</b>

Table 4.3: Performance of various models using full-length ROUGE metrics.

- **EARD-M<sub>2</sub>**: This baseline model is similar to the above model. Additionally, it also operates on the attention weights and the context vector described in the M2 model in Section 4.3.

### 4.5.1 Implementation Details

We used 80% of the data for training, 10% for validation, and 10% for testing. We create 10 such folds and report the average ROUGE-1, ROUGE-2, ROUGE-L scores across the 10 folds. The hyperparameters (batch size and GRU cell sizes) of all the models are tuned on the validation set. We tried the following batch sizes: {32, 64} and the following GRU cell sizes {200, 300, 400}. We used Adam (Kingma and Ba, 2014) as the optimization algorithm with the initial learning rate set to 0.0004,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ . We used pre-trained publicly available Glove word embeddings<sup>1</sup> and fine-tuned them during training. The same word embeddings are used for the query words and the passage words.

<sup>1</sup><http://nlp.stanford.edu/projects/glove/>

---

---

**Source:** Although cannabis does indeed have some harmful effects, it is no more harmful than legal substances like alcohol and tobacco. As a matter of fact, research by the British Medical Association shows that nicotine is far more addictive than cannabis. Furthermore, the consumption of alcohol and the smoking of cigarettes cause more deaths per year than does the use of cannabis (e.g. through lung cancer, stomach ulcers, accidents caused by drunk driving etc.). The legalization of cannabis will remove an anomaly in the law whereby substances that are more dangerous than cannabis are legal whilst the possession and use of cannabis remains unlawful.

**Query:** is marijuana harmless enough to be considered a medicine

**Ground Truth:** marijuana is no more harmful than tobacco and alcohol

---

*Generated Summaries:*

---

**EAD-Query:** marijuana is no the **drug drug** for **tobacco and tobacco**

**EARD-D<sub>1</sub>:** marijuana is no more harmful than tobacco and tobacco

**EARD-SD<sub>1</sub>:** marijuana is more for evidence than tobacco and health

**EARD-D<sub>2</sub>:** marijuana is no more harmful than tobacco and use

**EARD-SD<sub>2</sub>:** marijuana is no more harmful than tobacco and alcohol

---

**Source:** Fuel cell critics point out that hydrogen is flammable, but so is gasoline. Unlike gasoline, which can pool up and burn for a long time, hydrogen dissipates rapidly. Gas tanks tend to be easily punctured, thin-walled containers, while the latest hydrogen tanks are made from Kevlar. Also, gaseous hydrogen isn't the only method of storage under consideration—BMW is looking at liquid storage while other researchers are looking at chemical compound storage, such as boron pellets.

**Query:** safety are hydrogen fuel cell vehicles safe

**Ground Truth:** hydrogen in cars is less dangerous than gasoline

---

*Generated Summaries:*

---

**EAD-Query:** hydrogen is **hydrogen hydrogen hydrogen** fuel energy

**EARD-D<sub>1</sub>:** hydrogen in cars is less natural than gasoline

**EARD-SD<sub>1</sub>:** hydrogen in cars is reduce risk than fuel

**EARD-D<sub>2</sub>:** hydrogen in waste is less effective than gasoline

**EARD-SD<sub>1</sub>:** hydrogen in cars is less dangerous than gasoline

---

**Source:** The basis of all animal rights should be the Golden Rule: we should treat them as we would wish them to treat us, were any other species in our dominant position.

**Query:** do animals have rights that makes eating them inappropriate

**Ground Truth:** animals should be treated as we would want to be treated

---

*Generated Summaries:*

---

**EAD-Query :** animals should be **treated** as we would protect to be **treated**

**EARD-D<sub>1</sub>:** animals should be **treated** as we most individual to be **treated**

**EARD-SD<sub>1</sub>:** animals should be **treated** as we would physically to be treated

**EARD-D<sub>2</sub>:** animals should be **treated** as we would illegal to be **treated**

**EARD-SD<sub>2</sub>:** animals should be **treated** as those would want to be **treated**

---

Table 4.4: Summaries generated by different models. In general, we observed that the baseline models which do not use a diversity based attention model tend to produce more repetitions. Notice that the last example shows that our model is not very aggressive in dealing with the history and is able to produce valid repetitions (treated ... treated) when needed.

---

---

**Source:** Education is used to inculcate values that the society believes in and to promote social and civic awareness . Religious schools promoting anti-female policies for example should not be receiving taxpayer funding in the form of vouchers . This violates the constitutional separation of church and state . State funds can not be separated from state control . It matters not whether the state is not directly making the choice to fund religious schools . Taxpayer funds should not be allowed to be directed toward religious schools .

**Query:** Do vouchers rightly/wrongly support religious schools ?

**Ground Truth:** Vouchers for religious schools violates church / state separation

---

**EARD-SD<sub>2</sub>:** religious vouchers violates religious schools.

---

**Source:** The White House insists the treaty doesn't affect it but the Kremlin's takes a different view : "Can operate and be viable only if the United States of America refrains from developing its missile-defense capabilities quantitatively or qualitatively. "

**Query:** Does new start preserve missile defense capabilities ?

**Ground Truth:** New start restricts us missile defense options

---

**EARD-SD<sub>2</sub>:** the missile house treaty can be viable .

---

Table 4.5: Generated Samples where our best model, **EARD-SD<sub>2</sub>**;, does not restore the semantic meaning or does not generate the summary specific to the given query.

## 4.6 Results and Discussions

In this section, we discuss the results of the experiments reported in Table 4.3. Note that Row 12 is the current state-of-the-art result for this dataset, and it has been added for completeness. Su *et al.* (2021) is a BART based model proposed for the task of query focused abstractive summarization.

**1. Effect of Query:** Comparing rows 1 and 2 we observe that adding an encoder for the query and allowing it to influence the outputs of the decoder indeed improves the performance. This is expected as the query contains some keywords that could sharpen the summary's focus.

**2. Effect of Query attention model:** Comparing rows 2 and 3, we observe that using an attention model to compute the query representation at each time step improves the results. This suggests that the attention model indeed learns to focus on relevant portions of the query at different time steps.

**3. Effect of Diversity models:** All the diversity models introduced in the paper (rows 7, 8, 9, 10) significantly improve over the non-diversity models. In particular, the

Model	#summaries with repeated unigrams
<b>EAD-Query</b> <sub>attn</sub>	498
<b>EARD-SD</b> <sub>1</sub>	352
<b>EARD-SD</b> <sub>2</sub>	344
<b>EARD-D</b> <sub>1</sub>	191
<b>EARD-D</b> <sub>2</sub>	179

Table 4.6: The number of sentences with repeated unigrams across generated summaries from various models. We average out the total number of sentences across 10 folds. We can infer that the model with hard orthogonalization in Diversity LSTM generates summaries with least number of repetitions.

Diversity-LSTM model gives the best results. This is indeed very encouraging, and Table 4.4 shows some sample summaries comparing different models’ performance. Moreover, with beam search decoding with beam width as 5, we observe an additional 2% relative improvement in the performance over our best performing model.

**4. Comparison with baseline diversity models:** The baseline diversity model **EARD-M**<sub>1</sub> (row 5) performs at par with our models **EARD-D**<sub>1</sub> and **EARD-SD**<sub>1</sub> but not as good as **EARD-D**<sub>2</sub> and **EARD-SD**<sub>2</sub>. However, the model **EARD-M**<sub>2</sub> (row 6) performs very poorly. We believe that simultaneously adding a constraint on the context vectors and attention weights (as is indeed the case with **EARD-M**<sub>2</sub>) is a bit too aggressive. This could lead the model to diversify even when the models need to attend to similar content to complete the summary and thus could be leading to poor performance.

**5. Quantitative Analysis:** In addition to the qualitative analysis reported in Table 4.4 we also did a quantitative analysis by counting the number of sentences containing repeated words generated by different models. Specifically, for the 1268 test instances we counted the number of sentences containing repeated words as generated by different modes. Table 4.6 summarizes this analysis. We observed that the number of repeated words indeed decreases with our models. We also present a few samples where our model **EARD-SD**<sub>2</sub> does not generate semantically correct summary (Example 1 in Table 4.5). Also, there are instances where the generated summary is not specific to the given query (Example 2 in Table 4.5). This shows that there is still scope for improving systems for query based abstractive summarization.

## 4.7 Summary

In this work, we proposed a method for query-based abstractive summarization. The unique feature of the model is a novel diversification mechanism based on successive orthogonalization. This gives us the flexibility to (i) provide diverse context vectors at successive time steps and (ii) pay attention to words repeatedly if need be later in the summary (as opposed to existing models which aggressively delete the history). We also introduced a new dataset and empirically verified that we perform significantly better with a gain of 28% (absolute) in the ROUGE-L score when compared to a vanilla encode-attend-decode model. We observe that adding an attention mechanism to the query string gives significant improvements. We also compare with a state-of-the-art diversity model and outperform it with a gain of 7% (absolute) in the ROUGE-L score.

## CHAPTER 5

# Exploiting Task Specific Characteristics for improving Adequacy

RNN based sequence-to-sequence models with an attention network have been adopted for the task of structured data to text generation (Mei *et al.*, 2016). However, these models do not capture any of the task specific characteristics such as the necessity to cover all/most fields in a sequential order. In this chapter, we discuss some of these task specific characteristics and then propose a *refine* module to handle these characteristics.

### 5.1 Introduction

Rendering natural language descriptions from structured data is required in a wide variety of commercial applications, such as generating descriptions of products, hotels, furniture, *etc.*, from a corresponding table of facts about the entity. Such a table typically contains  $\{field, value\}$  pairs where the field is a property of the entity (*e.g.*, *color*) and the value is a set of possible assignments to this property (*e.g.*, *color = red*). Another example of this is the well-known task of generating one line biography descriptions from a given Wikipedia infobox (Lebret *et al.*, 2016). The Wikipedia infobox serves as a table of facts about a person, and the first sentence from the corresponding article serves as a one line description of the person. Figure 5.1 illustrates an example input infobox which contains fields such as  $\{Born, Position, Played For, National team, and Playing career\}$ . Each field further contains some words (*e.g.*, Defence, Canada, 1991-2006, Chicago Blackhawks, *etc.*). The corresponding description is coherent with the information contained in the infobox.

Note that the number of fields in the infobox and the ordering of the fields within the infobox vary from person to person. Given the large size (700K examples) and heterogeneous nature of the dataset, which contains biographies of people from different



<b>Karl Dykhuis</b>	
<b>Born</b>	July 8, 1972 (age 48) Sept-Îles, Quebec, Canada
<b>Height</b>	6 ft 3 in (191 cm)
<b>Weight</b>	214 lb (97 kg; 15 st 4 lb)
<b>Position</b>	Defence
<b>Shot</b>	Left
<b>Played for</b>	Chicago Blackhawks Philadelphia Flyers Tampa Bay Lightning Montreal Canadiens
<b>National team</b>	 Canada
<b>NHL Draft</b>	16th overall, 1990 Chicago Blackhawks
<b>Playing career</b>	1991–2006

Figure 5.1: The goal is to generate the following description when this infobox is passed as an input: *Karl Dykhuis born (born July 8, 1972) is a Canadian former professional ice hockey defenceman who played 12 seasons in the National Hockey League (NHL) for the Chicago Blackhawks, Philadelphia Flyers, Tampa Bay Lightning and Montreal Canadiens.*

backgrounds (sports, politics, arts, *etc.*), it is hard to come up with simple rule-based templates for generating natural language descriptions from infoboxes, thereby making a case for data-driven models. Based on the recent success of data-driven neural models for various other NLG tasks (Bahdanau *et al.*, 2014; Rush *et al.*, 2015; Yao *et al.*, 2015; Chopra *et al.*, 2016a), several works (Lebret *et al.*, 2016; Mei *et al.*, 2016; Liu *et al.*, 2019a) have adapted seq2seq models for this task. These models incorporate the tabular structure of the infobox by using a hierarchical encoder and/or attention network. However, as we observe from Table 5.1, the generated descriptions are still sometimes incoherent (Example 1) and misrepresent some of the information (Examples 2 and 3) present in the given input table. We hypothesize that is because such a model still does not exploit other crucial specific characteristics of this task, as explained below.

We observe that while rendering the output, once the model pays attention to a field (say, occupation), it needs to stay on this field for a few timesteps (till all the occupations are produced in the output). We refer to this as the *stay on* behavior. Further, we note that once the tokens of a field are referred to, they are usually not referred to later. For example, once all the occupations have been listed in the output, we will never revisit the occupation field because nothing is left to say about it. We refer to this as the *never*

Axelrod in 1970	
Background information	
<b>Born</b>	April 17, 1931 <sup>[1]</sup> Los Angeles, California, US
<b>Died</b>	February 5, 2017 (aged 85) Burbank, California, US <sup>[2]</sup>
<b>Genres</b>	Jazz, funk, soul <sup>[3]</sup>
<b>Occupation(s)</b>	Producer, arranger, composer

**Reference:** David Axelrod (born April 17 , 1933) is an American composer, arranger, and producer .

**Mei et al. (2016)** David Axelrod is an American composer , producer , arranger , **and arranger**

The Right Honourable <b>Sir Lester Bird</b> KNH		2nd Prime Minister of Antigua and Barbuda	
<b>Personal details</b>		<b>In office</b>	
<b>Born</b>	21 February 1938 (age 83) New York, New York, United States	9 March 1994 – 24 August 2004	
<b>Political party</b>	Labour	<b>Monarch</b>	Elizabeth II
<b>Alma mater</b>	University of Michigan	<b>Governor-General</b>	James Carlisle
		<b>Preceded by</b>	Vere Bird
		<b>Succeeded by</b>	Baldwin Spencer

**Reference:** Sir Lester Bryant Bird , KNH ( born 21 February 1938 ) was the second Prime Minister of Antigua and Barbuda from 1994 to 2004 and a well-known athlete .

**Mei et al. (2016):** Sir **James** Lester Bird ( born 21 February 1938 ) is a former politician from the **United States of Canada** .

<b>Brad Turner</b>	
<b>Born</b>	June 22 Bayfield, Ontario, Canada
<b>Occupation</b>	Film director, television director, television producer, photographer
<b>Years active</b>	1979–present

**Reference:** Brad Turner is a Canadian film director , television director and photographer .

**Mei et al. (2016):** Brad Turner is an **American** film and television director .

Table 5.1: Examples of generated descriptions from baseline model. The first example is incoherent. In the second example, the model hops between “Name” and “Governor-General” fields. For the last example, the *nationality* is incorrect generated by a baseline model is incorrect.

*look back* behavior.

In this chapter, we focus on these two specific characteristics (*stay-on* and *never-look-back*) behavior and model them in our refine module. To elaborate, we introduce a learnable forget (or remember) gate, which acts as a signal to decide when to forget the current field (or equivalently to decide till when to remember the current field). To model the *never look back* behavior, we introduce a gated orthogonalization mechanism which ensures that once a field is forgotten, subsequent field context vectors fed to the decoder are orthogonal to (or different from) the previous field context vectors.

We validate that our encode-attend-refine-decode paradigm is better than encode-attend-decode paradigm on the WIKIBIO dataset (Lebret *et al.*, 2016) which contains around 700K {infobox, description} pairs and has a vocabulary of around 400K words. We show that our model gives a relative improvement of 21% and 20% as compared to the current state-of-the-art models (Lebret *et al.*, 2016; Mei *et al.*, 2016) on this dataset. The proposed model also gives a relative improvement of 10% as compared to the basic seq2seq model. Further, we also show that our model outperforms state-of-the-art methods mentioned above on French and German WIKIBIO datasets proposed in Shetty M (2018).

## 5.2 Proposed model

As input we are given an infobox  $\mathcal{I} = \{(g_i, k_i)\}_{i=1}^N$ , which is a set of pairs  $(g_i, k_i)$  where  $g_i$  corresponds to *field* names and  $k_i$  is the sequence of corresponding *values* and  $N$  is the total number of fields in  $\mathcal{I}$ . For example,  $(g = \textit{occupation}, k = \textit{actor, writer, director})$  could be one such pair in this set. Given such an input, the task of generating a description  $\mathbf{y} = \{y_1, y_2, \dots, y_m\}$  for the given infobox can be modeled as the problem of finding a  $\mathbf{y}^*$  that maximizes the probability  $p(\mathbf{y}|\mathcal{I})$  which can be written as:

$$\mathbf{y}^* = \arg \max_{\mathbf{y}} \prod_{t=1}^m p(y_t | y_1, \dots, y_{t-1}, \mathcal{I}) \quad (5.1)$$

We now describe a way of modeling  $p(y_t | y_1, \dots, y_{t-1}, \mathcal{I})$  using the proposed *encode-attend-refine-decode* paradigm. The proposed model contains the following compo-

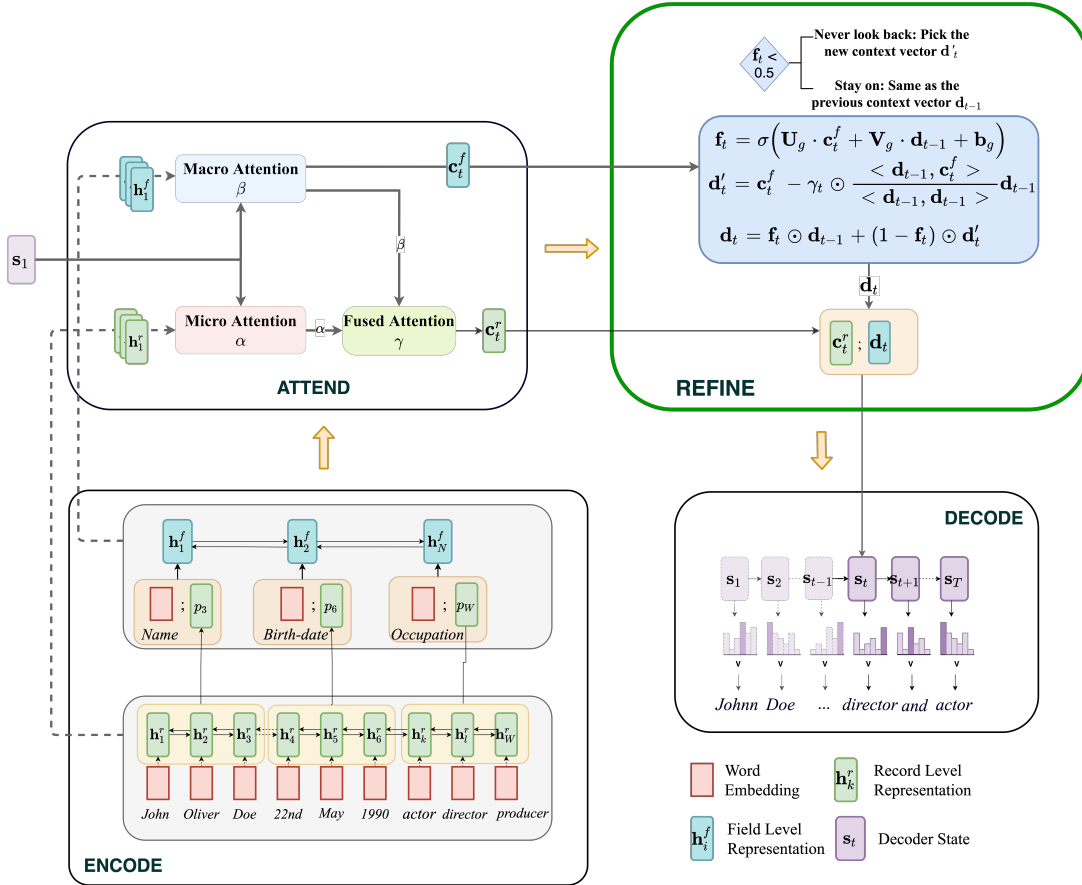


Figure 5.2: Our proposed model consists of i) a hierarchical encoder that encodes the infobox at the field and value level, ii) attend module with three sub-components: micro, macro, and fused attention networks, iii) refine module models the *stay-on* and *never-look-back* characteristics. Here  $\mathbf{d}_t$  represents the history of filed level context vectors, iv) a decoder that generates the description conditioned on value-level and refined field-level context vectors.

ments: (i) a hierarchical encoder RNN for the Infobox as described in Shetty M (2018) (iii) a bifocal attention mechanism which attends to information at the micro and macro level as described in Shetty M (2018) (iii) a refine module to modify the contextual representations and (iv) a decoder RNN to generate the output, one word at a time.

## 5.2.1 Encode

**Representation of a field:** Our goal is to compute a representation for the input, an infobox i.e., a table. As a table can be viewed as an aggregation of fields, the representation of the infobox can be derived by applying a function  $g$  over the field representations. Thus, we first focus on computing the representation for each field in the table.

Note that a field itself contains many values. For example, the field *occupation* for *Ben Affleck* may contain the values *actor*, *writer*, *director*. Hence, a field’s representation can either be (i) the word embedding of the field name or (ii) some function  $f$  of the the representations of the values in the field or a concatenation of (i) and (ii). We found that concatenating the representation of the field name with the aggregated representation of the values in the field works best. Further, using a bidirectional GRU cell to take contextual information from neighboring fields also helps. Hence, we compute the representation of the  $j$ -th field,  $\mathbf{h}_j^f$  as follows:

$$\mathbf{h}_j^f = \text{BiGRU}(\mathbf{h}_{i-1}^f, [\mathbf{e}(x_i^f) : \mathbf{h}^r]) \quad (5.2)$$

where  $\mathbf{e}(x_i^f)$  represents the word embedding for the field name,  $x_i^f$ , and  $\mathbf{h}^r$  is an aggregated representation of the values in the field. This aggregated representation of the values in the field is computed as described below.

**Representation of a value within a field:** Let  $\mathbf{h}_j^r$  be the representation of the  $j$ -th value in a given field. This representation could again either be (i) simply the embedding of this value (ii) or a contextual representation computed using a function  $f$ , which also considers the other values in the field. For example, if  $(x_1^r, x_2^r, \dots, x_l^r)$  are the values in a field, then these values can be treated as a sequence, and the representation of the  $j$ -th value can be computed using a bidirectional GRU over this sequence. Once again, we found that using a bidirectional GRU works better than simply using the embedding of the value. Thus, we compute the representation of the  $i$ -th value in the field as:

$$\mathbf{h}_i^r = \text{BiGRU}(\mathbf{h}_{i-1}^r, \mathbf{e}(x_i^r)) \quad (5.3)$$

Note that the representation  $\mathbf{h}^r$  in Equation (5.2) refers to the contextual representation of the last value  $\mathbf{h}_l^r$  in the sequence (which in turn contains information from all the other values in the field).

## 5.2.2 Attend

While understanding and generating a description from structured data, a human would keep track of information at two levels. Specifically, at a macro level, she would first decide which field to mention next and then at a micro level, decide which of the values in the field needs to be mentioned next. For example, she first decides that the field *occupation* needs attention at the current step. She then decides which is the next appropriate occupation to attend to from the set of occupations (*actor, director, producer, etc.*). To enable this, we use a hierarchical encoder and a bifocal attention mechanism which computes attention over *fields* at a macro level and *values* at a micro level. We then fuse these attention weights such that the attention weight for a field also influences the attention over the values within it, as discussed below. This idea of fusing the macro level attention with a micro level attention as described below is the same as that proposed in Shetty M (2018).

**Macro Attention:** Let  $\mathbf{h}_j^f$  be the representation of the  $j^{\text{th}}$  field in the infobox. Given these representations  $\{\mathbf{h}_j^f\}_{j=1}^N$  for all the  $N$  fields we compute an attention over the fields (macro level).

$$b_{t,j}^f = \mathbf{v}_f^T \cdot \tanh(\mathbf{U}_f \cdot \mathbf{s}_{t-1} + \mathbf{V}_f \cdot \mathbf{h}_j^f) \quad (5.4)$$

$$\beta_{t,j} = \frac{\exp(b_{t,j}^f)}{\sum_{i=1}^N \exp(b_{t,i}^f)} \quad (5.5)$$

$$\mathbf{c}_t^f = \sum_{i=1}^N \beta_{t,i} \mathbf{h}_i^f \quad (5.6)$$

where  $\mathbf{s}_{t-1}$  is the state of the decoder at time step  $t - 1$ .  $\mathbf{U}_f$ ,  $\mathbf{V}_f$  and  $\mathbf{v}_f$  are parameters,  $N$  is the total number of fields in the input and  $\mathbf{c}_t^f$  is the macro (field level) context vector at the  $t$ -th time step of the decoder.

**Micro Attention:** Once we have computed a representation for all values across all the fields as described earlier, we compute the attention over these values (micro level) as

shown below :

$$a_{t,i}^r = \mathbf{v}_r^T \cdot \tanh(\mathbf{U}_r \cdot \mathbf{s}_{t-1} + \mathbf{V}_r \cdot \mathbf{h}_i^r) \quad (5.7)$$

$$\alpha_{t,i}^r = \frac{\exp(a_{t,i}^r)}{\sum_{i=1}^W \exp(a_{t,i}^r)} \quad (5.8)$$

where  $\mathbf{s}_{t-1}$  is the state of the decoder at time step  $t - 1$ .  $\mathbf{U}_r$ ,  $\mathbf{V}_r$  and  $\mathbf{v}_r$  are parameters,  $W$  is the total number of values across all the fields.

**Fused Attention:** Intuitively, the attention weights assigned to a field should have an influence on all the values belonging to the particular field. To ensure this, we reweigh the micro-level attention weights based on the corresponding macro-level attention weights. In other words, we fuse the attention weights at the two levels as:

$$\alpha'_{t,i} = \frac{\alpha_{t,i} \beta_{t,F(i)}}{\sum_{j=1}^W \alpha_{t,j} \beta_{t,F(j)}} \quad (5.9)$$

$$\mathbf{c}_t^r = \sum_{j=1}^W \alpha'_{t,i} \mathbf{h}_i^r \quad (5.10)$$

where  $F(j)$  is the field corresponding to the  $i$ -th value and  $\mathbf{c}_t^r$  is the macro level context vector.

### 5.2.3 Refine

We now describe the design of the refine module to model the *stay-on* and *never-look-back* behavior.

**Stay-On:** We first begin with the *stay-on* property, which essentially implies that if we have paid attention to the field  $i$  at timestep  $t$ , we are likely to pay attention to the same field for a few more time steps. For example, suppose we are focusing on the *occupation* field at this timestep. In that case, we tend to focus on it for the next few timesteps until all relevant values in this field have been included in the generated description. In other words, we want to remember the field context vector  $\mathbf{c}_t^f$  for a few timesteps. One way of ensuring this is to use a remember (or forget) gate as given below, which remembers the

previous context vector when required and forgets it when moving on from that field.

$$\mathbf{f}_t = \sigma(\mathbf{U}_g \cdot \mathbf{c}_t^f + \mathbf{V}_g \cdot \mathbf{d}_{t-1} + \mathbf{b}_g) \quad (5.11)$$

$$\mathbf{d}_t = (1 - \mathbf{f}_t) \odot \mathbf{c}_t^f + \mathbf{f}_t \odot \mathbf{d}_{t-1} \quad (5.12)$$

where  $\mathbf{U}_g, \mathbf{V}_g, \mathbf{b}_g$  are parameters to be learned. The job of the forget gate is to ensure that  $\mathbf{c}_t$  is similar to  $\mathbf{c}_{t-1}$  when required (*i.e.*, by learning  $\mathbf{f}_t \rightarrow \mathbf{1}$  when we want to continue focusing on the same field) and different when it is time to move on (by learning that  $\mathbf{f}_t \rightarrow \mathbf{0}$ ).

**Never look back:** Next, the *never look back* property implies that once we have moved away from a field, we are unlikely to pay attention to it again. For example, once we have rendered all the occupations in the generated description, there is no need to return to the occupation field. In other words, once we have moved on ( $\mathbf{f}_t \rightarrow \mathbf{0}$ ), we want the successive field context vectors to be very different from the previous field vectors. One way of ensuring this is to orthogonalize successive field vectors using:

$$\mathbf{d}'_t = \mathbf{c}_t^f - \gamma_t \odot \frac{\langle \mathbf{d}_{t-1}, \mathbf{c}_t^f \rangle}{\langle \mathbf{d}_{t-1}, \mathbf{d}_{t-1} \rangle} \mathbf{d}_{t-1} \quad (5.13)$$

where  $\langle \mathbf{a}, \mathbf{b} \rangle$  is the dot product between vectors  $\mathbf{a}$  and  $\mathbf{b}$ . The above equation essentially subtracts the component of  $\mathbf{c}_t^f$  along  $\mathbf{d}_{t-1}$ .  $\gamma_t$  is a learned parameter that controls the degree of orthogonalization, thereby allowing a soft orthogonalization (*i.e.*, the entire component along  $\mathbf{d}_{t-1}$  is not subtracted but only a fraction of it). The above equation only ensures that  $\mathbf{c}_t^f$  is soft-orthogonal to  $\mathbf{d}_{t-1}$ . Alternately, we could pass the sequence of context vectors,  $\{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_t\}$  generated so far through a GRU cell. The state of this GRU cell at each time step would thus be aware of the history of the field vectors till that timestep. Now instead of orthogonalizing  $\mathbf{c}_t^f$  to  $\mathbf{d}_{t-1}$  we could orthogonalize  $\mathbf{c}_t^f$  to the hidden state of this GRU at time-step  $t - 1$ . We found this to work better in practice as it accounts for all the field vectors in the history instead of only the previous field vector.

In summary, Equation (5.12) provides a mechanism for remembering the current field vector when appropriate (thus capturing *stay-on* behavior) using a remember gate. On the other hand, Equation 5.13 explicitly ensures that the field vector is very different



(soft-orthogonal) from the previous field vectors once it is time to move on (thus capturing *never look back* behavior). The value of  $\mathbf{d}'_t$  computed in Equation (5.13) is then used in Equation (5.12) in place of  $\mathbf{c}_t^f$ . More formally the stay-on and never-look-back properties are combined as:

$$\mathbf{d}_t = (1 - \mathbf{f}_t) \odot \mathbf{d}'_t + \mathbf{f}_t \odot \mathbf{d}_{t-1} \quad (5.14)$$

, where  $\mathbf{f}_t$  is the remember gate introduced in Equation (5.11). The  $\mathbf{d}_t$  (macro) thus obtained is then concatenated with  $\mathbf{c}_t^r$  (micro) and fed to the decoder (refer to Figure 5.2).

## 5.2.4 Decode

The hidden state of the decoder  $\mathbf{s}_t$  at each time  $t$  is computed using a GRU as follows:

$$\mathbf{s}_t = \text{GRU}(\mathbf{s}_{t-1}, [\mathbf{e}(y_{t-1}); \mathbf{d}_{t-1}; \mathbf{c}_{t-1}^r]) \quad (5.15)$$

where,  $\mathbf{e}(y_{t-1})$  is the  $d$ -dimensional of the word  $y_t$ , which has the highest probability under the following distribution over the vocabulary words at timestep  $t - 1$  computed as:

$$\tilde{p}(y_t) = \text{softmax}(\mathbf{W}_o \cdot f(\mathbf{W}_{dec} \cdot \mathbf{s}_t + \mathbf{V}_{dec} \cdot \mathbf{d}_t + \mathbf{U}_{dec} \cdot \mathbf{c}_t^r)) \quad (5.16)$$

where  $\mathbf{W}_o \in \mathbb{R}^{V \times l_1}$ ,  $\mathbf{W}_{dec} \in \mathbb{R}^{l_1 \times l_1}$ ,  $\mathbf{V}_{dec} \in \mathbb{R}^{l_1 \times l_2}$ ,  $\mathbf{U}_{dec} \in \mathbb{R}^{l_1 \times l_3}$ ,  $l_1$  is the size of decoder's hidden state,  $l_2$  is the size of  $\mathbf{d}_t$ ,  $l_3$  is the size of  $\mathbf{c}_t^r$ ,  $V$  is the vocabulary size. The above output is exactly the quantity defined in Equation (5.1) that we wanted to model  $(p(y_t|y_1, \dots, y_{t-1}, \mathcal{I}))$ . Also  $[\mathbf{e}(y_{t-1}), \mathbf{d}_{t-1}, \mathbf{c}_{t-1}^r]$  means a concatenation of the vectors  $\mathbf{e}(y_{t-1}), \mathbf{d}_{t-1}, \mathbf{c}_{t-1}^r$ . We chose  $f$  to be the identity function.

## 5.2.5 Copying Mechanism

To deal with the large vocabulary ( $\sim 400\text{K}$  words) due to the presence of named entities, we use a copying mechanism as a post-processing step. Specifically, we identify

Language	Train	Test	Valid	Vocabulary Size	Average sentence length
English	582,659	72,831	72,831	426,910	26.11
German	44,064	5,518	5,403	143,108	32.29
French	141,524	18,101	18,049	297,130	36.49

Table 5.2: Dataset Statistics for Wikipedia Infoboxes for generating descriptions in various languages. Note that the structured data is also present in the corresponding language.

the time steps at which the decoder produces unknown words (denoted by the special symbol UNK). For each such time step, we look at the attention weights on the input words and replace the UNK word by that input word that has received the maximum attention at this timestep. This process is similar to the one described in Luong *et al.* (2015). Even Lebret *et al.* (2016) have a copying mechanism tightly integrated with their model.

## 5.3 Experimental setup

In this section we describe our experimental setup.

### 5.3.1 Datasets

We use the WIKIBIO dataset introduced in Lebret *et al.* (2016) It consists of 728, 321 biography articles from English Wikipedia. A biography article corresponds to a person (sportsman, politician, historical figure, actor, *etc.*) Each Wikipedia article has an accompanying infobox which serves as the structured input, and the task is to generate the first sentence of the article (which typically is a one-line description of the person). We used the same train, valid, and test sets made publicly available by Lebret *et al.* (2016). We also use a similar dataset in French and German language containing biographies and infoboxes from Wikipedia (Shetty M, 2018). The dataset statistics are presented in Table 5.2. For German and French Wikibio datasets, we use the same mechanism to split the data into train, valid, and test split as mentioned in Lebret *et al.* (2016). Note that the English dataset is better in terms of the number of samples and the

vocabulary size (400K) in comparison to German (143K) and French (297K) datasets. However, the descriptions are longer in the German and French datasets as compared to the English dataset.

### 5.3.2 Models compared

We compare our *encode-attend-refine-decode* model with the following models:

**1. Lebret *et al.* (2016)** is a conditional language model which uses a feed-forward neural network to predict the next word in the description conditioned on *local characteristics* (*i.e.*, words within a field) and *global characteristics* (*i.e.*, overall structure of the infobox).

**2. Mei *et al.* (2016)** was proposed in the context of the WEATHERGOV and ROBOCUP datasets, which have a much smaller vocabulary. They use an improved attention model with additional regularizer terms, that influence the weights assigned to the fields.

**3. Basic EAD:** This is the vanilla encode-attend-decode model (Bahdanau *et al.*, 2014) as described in Chapter 2. Here, the infobox is flattened to produce the following input sequence (the words in bold are field names that act as delimiters).

*[Name] John Doe [Birth\_Date] 19 March 1981 [Nationality] Indian .....*

**4. Hierarchical EAD:** The above encode-attend-decode paradigm does not take the input structure into account while encoding the infobox or generating the description. To strengthen our baseline, we use the encode and attend modules as described in Sections 5.2.1 and 5.2.2. We refer to this as the hierarchical encode-attend-decode model.

### 5.3.3 Hyperparameter tuning

We tuned the hyperparameters of all the models using a validation set. As mentioned earlier, we used a bidirectional GRU cell as the function  $f$  for computing the representation of the fields and values (see Section 5.2.1). For all the models, we experimented with GRU state sizes of 128, 256, and 512. The total number of unique words in the corpus is around 400K (this includes the words in the infobox and the descriptions).

Of these, we retained only the top 20K words in our vocabulary (same as Lebret *et al.* (2016)). We initialized the embeddings of these words with 300 dimensional Glove embeddings (Pennington *et al.*, 2014). Based on the validation set results, we found that fine-tuning the word embeddings does not benefit our final models, so we did not fine-tune the word embeddings. We used Adam (Kingma and Ba, 2014) with a learning rate of 0.0004,  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . We trained the model for a maximum of 20 epochs and used early stopping with the patience set to 5 epochs. We also use beam search with the beam size set to 5.

## 5.4 Results and Discussions

In this section, we discuss the results and the main observations from our experiments.

### 5.4.1 Comparison of different models

Model	BLEU-4	NIST-4	ROUGE-4
Lebret <i>et al.</i> (2016)	34.70	7.98	25.80
Mei <i>et al.</i> (2016)	35.10	7.27	30.90
Basic EAD	38.20	8.47	34.28
Hierarchical EAD	41.22	8.96	38.71
<i>Ours</i> (EARD)	42.03	9.17	39.11
+Beam Search (k=5)	<b>42.97</b>	<b>9.26</b>	<b>40.24</b>
<b>SOTA</b> Liu <i>et al.</i> (2017b)	<b>44.89</b>	-	<b>41.21</b>

Table 5.3: Comparison of different models on the English WIKIBIO dataset.

Following Lebret *et al.* (2016), we used BLEU-4, NIST-4 and ROUGE-4 as the evaluation metrics as described in Section 2.7. We first make a few observations based on the results on the English dataset (Table 5.3). The basic encode-attend-decode model and the model proposed by Mei *et al.* (2016) perform better than the model proposed by Lebret *et al.* (2016). Our final model with the refine module gives the best performance and does 10% (relative) better than the closest baseline (basic seq2seq) and 21% (relative) better than the state of the art results (Lebret *et al.*, 2016). Note that the last row in Table 5.3 represents the current state of the art result for this dataset. In Table 5.4, we show some qualitative examples of the output generated by different models.

---

---

**Reference:** Samuel Smiles (23 December 1812 — 16 April 1904), was a Scottish author and government reformer who campaigned on a Chartist platform.

**Basic EAD:** samuel smiles (23 december 1812 – 16 april 1904) was an english books and author.

**Hierarchical EAD:** samuel smiles (23 december 1812 - 16 april 1904) was a british books and books.

**EARD:** samuel smiles (23 december 1812 - 16 april 1904) was a british biographies and author.

---

---

**Reference:** Thomas Tenison (29 September 1636 — 14 December 1715) was an English church leader, Archbishop of Canterbury from 1694 until his death.

**Basic EAD:** thomas tenison (14 december 1715 - 29 september 1636) was an english roman catholic archbishop.

**Hierarchical EAD:** thomas tenison (29 september 1636 - 14 december 1715) was an english clergyman of the roman catholic church.

**EARD:** thomas tenison (29 september 1636 - 14 december 1715) was archbishop of canterbury from 1695 to 1715.

---

---

**Reference:** Guy F. Cordon (April 24, 1890 — June 8, 1969) was a U.S. politician and lawyer from the state of Oregon.

**Basic EAD:** charles l. mcnary (april 24 , 1890 8 , 1969) was a united states senator from oregon.

**Hierarchical EAD:** guy cordon (april 24 , 1890 – june 8 , 1969) was an american attorney and politician.

**EARD:** guy cordon (april 24 , 1890 – june 8 , 1969) was an american attorney and politician from the state of oregon.

---

---

**Reference:** Dr. Harrison B. Wilson Jr. (born April 21, 1925) is an American educator and college basketball coach who served as the second president of Norfolk State University from 1975-1997.

**Basic EAD:** lyman beecher brooks (born april 21 , 1925) is an american educator and educator.

**Hierarchical EAD:** harrison b. wilson , jr. (born april 21 , 1925) is an american educator and academic administrator.

**EARD:** harrison b. wilson , jr. (born april 21 , 1925) is an american educator , academic administrator , and former president of norfolk state university.

---

---

Table 5.4: Examples of generated descriptions from different models. For the last two examples, *name* generated by the basic seq2seq model is incorrect because it attended to the *preceded by* field.

## 5.4.2 Human Evaluation

Metric	A < B	A == B	A > B
<b>Adequacy</b>	186	208	106
<b>Fluency</b>	244	108	148
<b>Preference</b>	207	207	86

Table 5.5: Qualitative Comparison of Model A (Seq2Seq) and Model B (our model).

To make a qualitative assessment of the generated sentences, we conducted a human study on 500 Infoboxes sampled from the English dataset. The annotators for this task were undergraduate and graduate students. For each of these infoboxes, we generated summaries using the baseline **EAD** model and our proposed **EARD** model. For each description and for each model, we asked three annotators to rank the output of the systems based on i) adequacy (*i.e.* does it capture relevant information from the infobox), (ii) fluency (*i.e.* grammar), and (iii) relative preference (*i.e.*, which of the two outputs would be preferred). The average fluency/adequacy (on a scale of 5) was 4.04/3.6 and for the **EAD** model and 4.19/3.9 for our model. Note that the 5-point scale is the standard scaling as used in Papineni *et al.* (2002) where 1 maps to (very bad) and 5 maps to (very good) for both fluency and adequacy. We computed Fleiss Kappa score to compute the inter annotator agreement among the three models. We achieved moderate agreement with Kappa score of 0.62 and 0.65 for fluency and adequacy scores.

The results from Table 5.5 suggest that in general our **EARD** model performs better than the **EAD** model. Additionally, annotators were asked if the generated summaries look natural (*i.e.*, as if humans generated them). In 423 out of 500 cases, the annotators said “Yes” suggesting that **EARD** model indeed produces good descriptions.

## 5.4.3 Performance on different languages

The results on the French and German datasets are summarized in Tables 5.6 and 5.7 respectively. Note that the code of Lebrecht *et al.* (2016) is not publicly available, hence we could not report numbers for French and German using their model. We observe that our final model gives the best performance - although the Hierarchical-**EAD** model performs poorly compared to the basic seq2seq model on French. However, the overall

<b>Model</b>	<b>BLEU-4</b>	<b>NIST-4</b>	<b>ROUGE-4</b>
Mei <i>et al.</i> (2016)	10.40	2.51	7.81
<b>EAD</b>	14.50	3.02	12.22
Hierarchical <b>EAD</b>	13.80	2.86	12.37
Ours <b>EARD</b>	15.52	3.30	12.80
+ Beam Search (k=5)	<b>16.21</b>	<b>3.51</b>	<b>13.23</b>

Table 5.6: Comparison of different models on the French WIKIBIO dataset.


<b>Model</b>	<b>BLEU-4</b>	<b>NIST-4</b>	<b>ROUGE-4</b>
Mei <i>et al.</i> (2016)	9.30	2.23	5.85
<b>EAD</b>	17.05	3.09	12.16
Hierarchical- <b>EAD</b>	20.38	3.43	14.89
Ours <b>EARD</b>	23.33	4.24	16.40
+ Beam Search (k=5)	<b>23.91</b>	<b>4.47</b>	<b>17.11</b>

Table 5.7: Comparison of different models on the German WIKIBIO dataset.

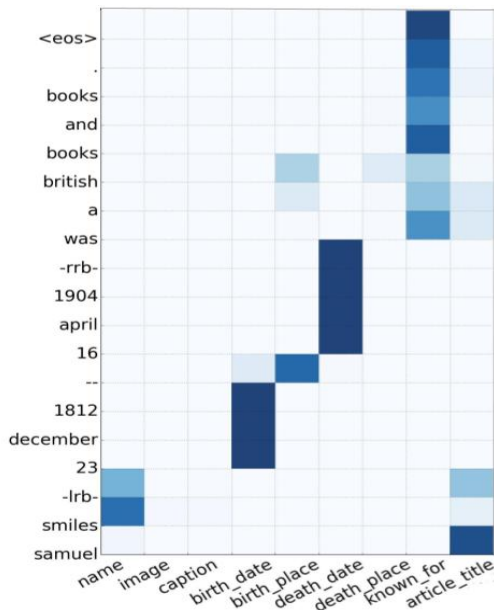
performance for French and German is much lower than that for English. There could be multiple reasons for this. First, the amount of training data in these two languages is smaller than that in English. Specifically, the amount of training data available in French (German) is only 24.2 (7.5)% of that available for English. Second, on average, the descriptions in French and German are longer than that in English (EN: 26.0 words, FR: 36.5 words, and DE: 32.3 words). Finally, a manual inspection across the three languages suggests that the English descriptions have a more consistent structure than French ones. For example, most English descriptions start with *name* followed by *date of birth*, but this is not the case in French. However, this is only a qualitative observation, and it is hard to quantify this characteristic of the French and German datasets.

#### 5.4.4 Visualizing Attention Weights

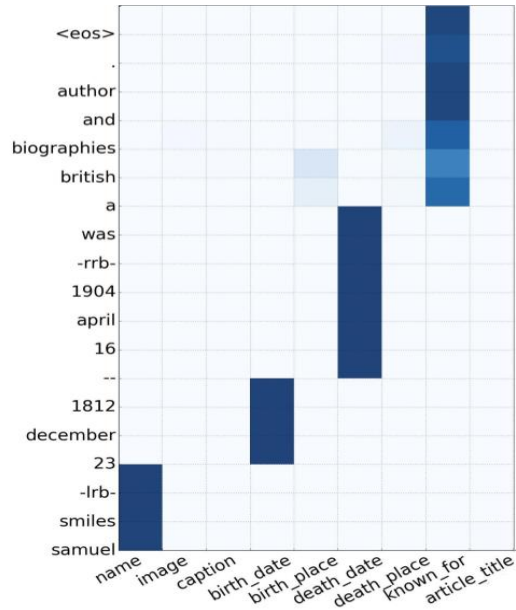
If the proposed model indeed works well, we should see attention weights consistent with the *stay on* and *never look back* behavior. To verify this, we plotted the attention weights in cases where our **EARD** model does better than the Hierarchical **EAD** model. Figure 5.3 shows the attention weights corresponding to the infobox in Figure 5.3a. Notice that the **EAD** model has attention on both the *name* field and the *article title* field while rendering the name. Our **EARD** model, on the other hand, stays on the name field for as long as it is required but then moves and never returns to it (as expected). Along with improving the generated descriptions, the crisp attention boundaries among

<b>Samuel Smiles</b>	<b>Born</b>	23 December 1812 <a href="#">Haddington, East Lothian, Scotland</a>
	<b>Died</b>	16 April 1904 (age 91) <a href="#">Kensington, London, England</a>
	<b>Known for</b>	Biographies and self-help books
	<b>Notable work</b>	<i>Self-Help</i>

(a) Wikipedia Infobox for Samuel Smiles



(b) Without Refine Module (EAD)



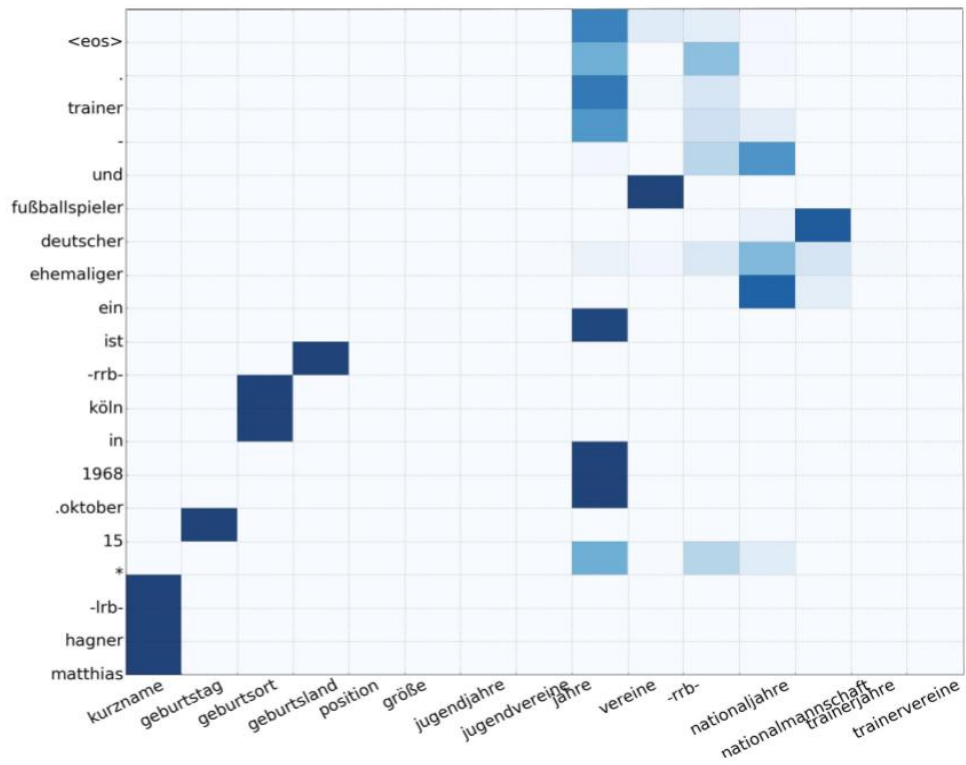
(c) With Refine Module (EARD)

Figure 5.3: Comparison of the attention weights and descriptions produced for Infobox in Figure 5.3a.

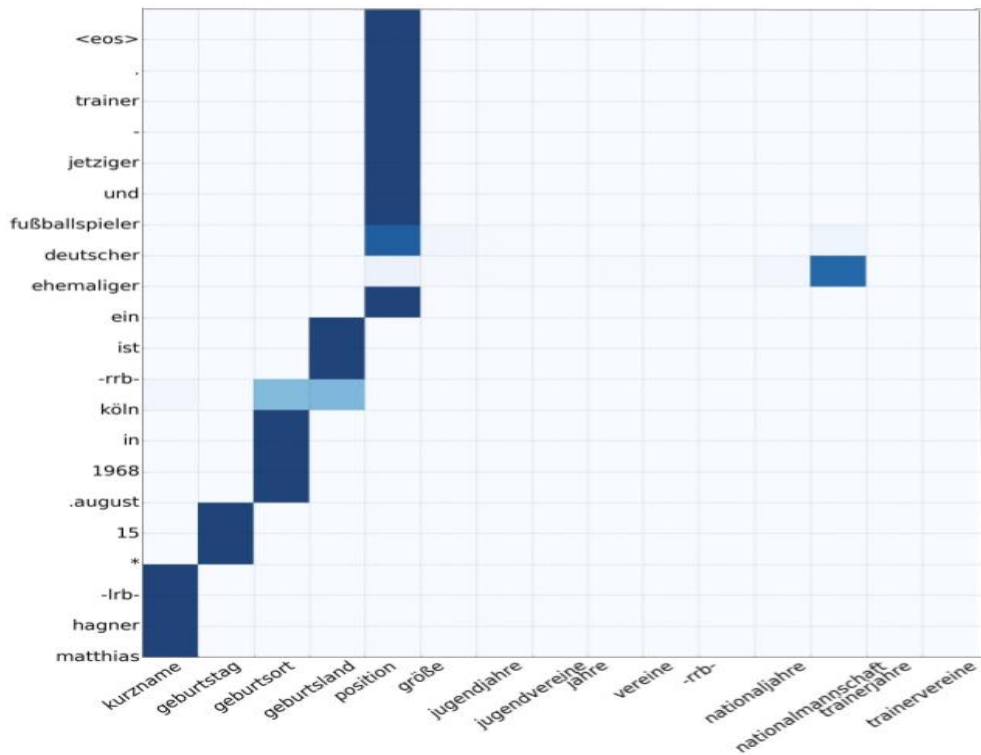
Matthias Hagner		Herren		
Personalia		Jahre	Station	Spiele (Tore) <sup>1</sup>
<b>Geburtstag</b>	15. August 1974	1991–1992	FC Burgsolms	
<b>Geburtsort</b>	Gießen, Deutschland	1992–1996	Eintracht Frankfurt Amat.	52 (22)
<b>Größe</b>	186 cm	1992–1996	Eintracht Frankfurt	31 (10)
<b>Position</b>	Mittelfeld	1996–1998	VfB Stuttgart	53 (10)
<b>Junioren</b>		1998–2001	Borussia Mönchengladbach	32 (2)
<b>Jahre</b>	<b>Station</b>	2001–2002	SpVgg Greuther Fürth	10 (1)
1979–1990	FC Burgsolms	2003	FSV Frankfurt	12 (7)
1990–1991	Eintracht Frankfurt	2003–2006	1. FC Saarbrücken	82 (17)
		2006–2009	FSV Frankfurt	64 (7)
		2009–2010	Eintracht Wetzlar	46 (16)

Figure 5.4: Wikipedia Infobox for Matthias Hagner.





(a) Without Refine Module (EAD)



(b) With Refine Module (EARD)

Figure 5.5: Comparison of the attention weights and descriptions in German produced for Infobox in Figure 5.4.

fields help in better understanding which words are being attended to while generating each output word.


We observed a similar trend for German and French also. As an example, we plotted the attention weights for the infobox in Figure 5.4 for German. We observe that while predicting the birth-date, the hierarchical EAD pays more attention to “jahre (years)”, which leads to a wrong birth-month in the output. On the contrary, our model **EARD** pays attention to the *birth-date* field consistently until the whole birth date is generated.

### 5.4.5 Out of domain results

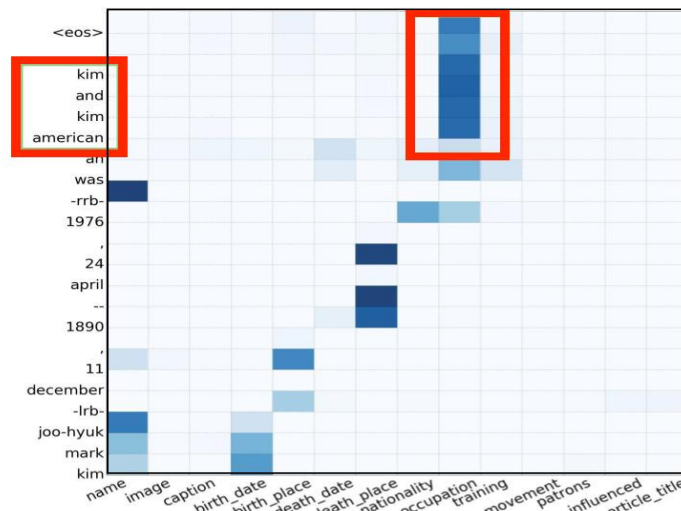
Training data	Performance on Target (test) data	
	Arts	Sports
Entire dataset	<b>33.6</b>	<b>52.4</b>
Without target domain data	24.5	29.3
+5k target domain data	31.2	41.8
+10k target domain data	32.2	43.3

Table 5.8: In this table, we focus on the shift in the performance when the model does not see any training samples from the corresponding domain, to samples being added gradually. We report the BLEU-4 scores for Wikipedia Infoboxes from two domains: Arts and Sports. We can infer that without the target domain samples the performance drops significantly. However, the performance recovers quickly through fine-tuning the model on 5K samples.

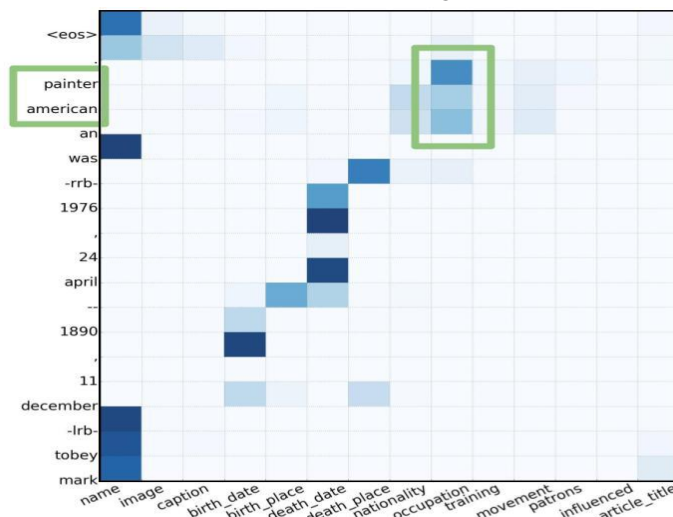
What if the model sees a different *type* of a person at test time? For example, what if the training data does not contain any sportspersons, but we encounter a sportsperson’s infobox at test time. This instance is the same as seeing out-of-domain data at test time. Such a situation is expected in the e-commerce domain, where new products with new features (fields) get frequently added to the catalog. We were interested in three questions here. First, we wanted to see if testing the model on out-of-domain data leads to a drop in performance. For this, we compared the performance of our best model in two scenarios (i) trained on data from all domains (including the target domain) and tested on the target domain (sports, arts) and (ii) trained on data from all domains except the target domain and tested on the target domain. Comparing rows 1 and 2 of Table 5.8 we observed a significant drop in the performance. Note that the numbers for the *Sports* domain in row 1 are much better than the *Arts* domain because roughly 40% of the WIKIBIO training data contains sportspersons.

	<b>Mark Tobey</b>	
	<b>Born</b>	December 11, 1890 Centerville, Wisconsin
	<b>Died</b>	April 24, 1976 (aged 85) Basel, Switzerland
	<b>Nationality</b>	American
	<b>Education</b>	School of the Art Institute of Chicago
	<b>Known for</b>	Painting
	<b>Movement</b>	Abstract Expressionism Northwest School
	<b>Patron(s)</b>	Zoe Dusanne

(a) Wikipedia Infobox for Mark Tobey.



(b) Without fine tuning.

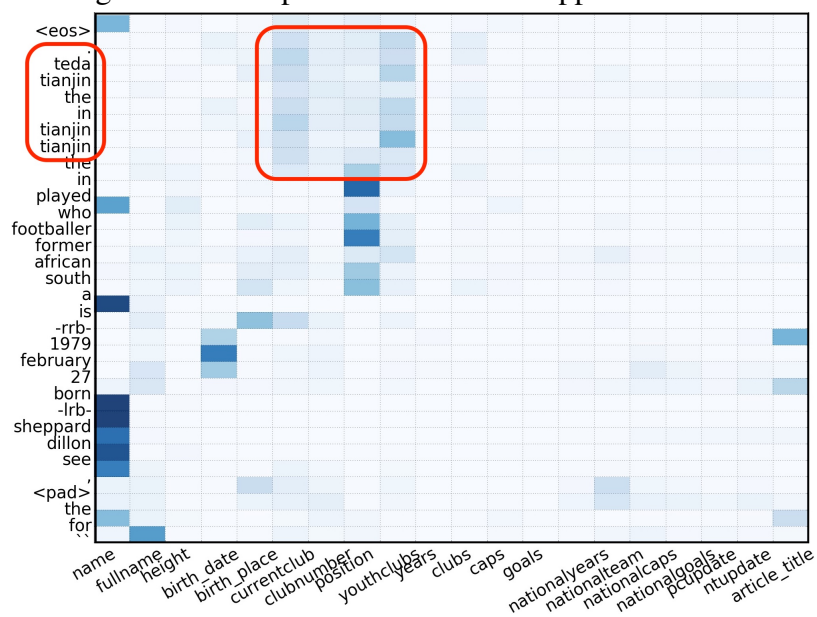


(c) With fine tuning with 5K in-domain data (Arts).

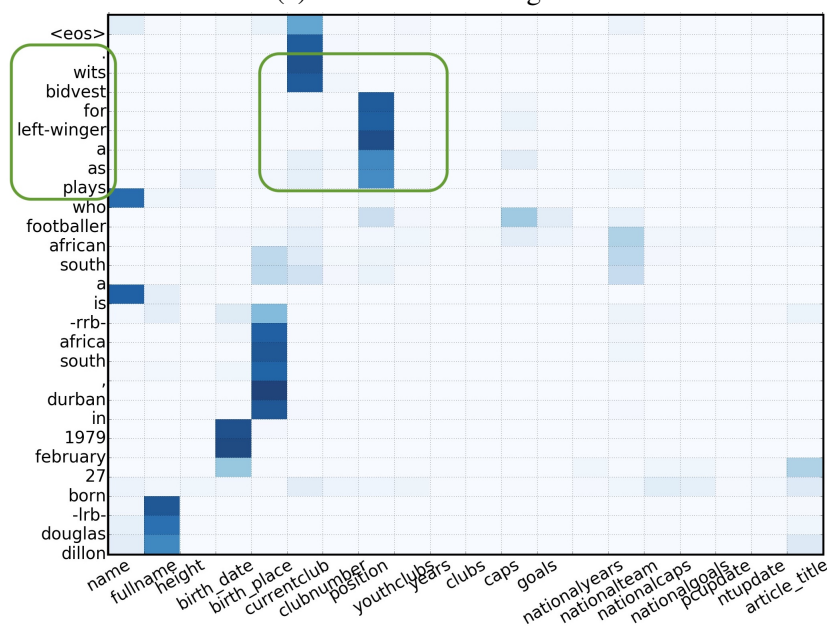
Figure 5.6: Comparison of the attention weights and descriptions (see highlighted boxes) produced by an out-of-domain model with and without fine tuning for the Infobox in Figure 5.6a .

Dillon Sheppard		Senior career*	
<b>Personal information</b>		<b>Years</b>	<b>Team</b>
<b>Full name</b>	Dillon Sheppard	1997–1999	Seven Stars
<b>Date of birth</b>	27 February 1979 (age 42)	1999–2004	Ajax Cape Town
<b>Place of birth</b>	Durban, South Africa	2004	Dynamo Moscow
<b>Height</b>	1.80 m (5 ft 11 in)	2005	Panionios
<b>Position(s)</b>	Left winger	2006–2009	Mamelodi Sundowns
<b>Youth career</b>		2009–2011	Platinum Stars
	[School of Excellence][Barcelona Youth]	2011–2013	Golden Arrows
<b>Senior career*</b>		2013–2014	Maritzburg United
		2014–2017	Bidvest Wits
			<b>Apps</b> (Gls)
			24 (1)
			105 (15)
			4 (0)
			10 (0)
			37 (1)
			39 (3)
			33 (4)
			14 (9)
			20 (2)

Figure 5.7: Wikipedia infobox for Sheppard Dillon.



(a) Without fine tuning.



(b) With fine-tuning with 5K in-domain data (Sports).

Figure 5.8: Comparison of the attention weights and descriptions (see highlighted boxes) produced by an out-of-domain model with and without fine-tuning for the Wikipedia infobox in Figure 5.7.

Next, we wanted to see if we can use a small amount of data from the target domain to fine-tune a model trained on the out-of-domain data. We observe that even with minimal amounts of target domain data, the performance improves significantly (see rows 3 and 4 of Table 5.8). Note that if we train a model from scratch with only limited data from the target domain instead of fine-tuning a model trained on a different source domain, the performance is very poor. In particular, training a model from scratch with 10K training instances, we get a BLEU score of 16.2 and 28.4 for arts and sports, respectively. Finally, even though the actual words used for describing a sportsperson (footballer, cricketer, *etc.*) would be very different from the words used to describe an artist (actor, musician, *etc.*) they might share many fields (for example, date of birth, occupation, *etc.*). As seen in Figure 5.6 (attention weights corresponding to the infobox in Figures 5.6a), the model predicts the attention weights correctly for common fields (such as name, occupation). However, it is unable to use the right vocabulary to describe the occupation for Figure 5.6 (since it has not seen such words frequently in the training data). Once we fine-tune the model with limited data from the target domain, we see that it picks up the new vocabulary and produces an accurate description of the occupation. Furthermore, we also illustrate in Figure 5.8 for the Sports domain, that with fine-tuning, our model is more confident in attending to the new relevant *fields* specific to this domain.

## 5.5 Summary

In this chapter, we described a model for generating natural language descriptions from structured data. To incorporate task-specific characteristics such as, *stay-on* and *never-look-back* behaviors, we propose a refine module which contains use gated orthogonalization to refine the context vectors generated at each timestep. We also use a hierarchical encoder with bifocal attention to capture the hierarchical structure in the input. Our final model outperforms an existing state-of-the-art model on a large scale WIKI-BIO dataset by 21%. We also demonstrate that our model gives state-of-the-art results on the French and German WikiBio datasets. Finally, we perform experiments with an out-of-domain model and show that if such a model is fine-tuned with small amounts of in-domain data, it can improve the target domain.

## CHAPTER 6

# Designing Task Specific Metric for improving Answerability

In this chapter, we discuss our contributions for the task of Automatic Question Generation (AQG). Current AQG systems based on the encode-attend-decode paradigm do not explicitly ensure that the systems generate *answerable* questions. We first discuss our proposed metric, which captures answerability. We then discuss our refine module, which uses this as a reward signal to generate better questions.

### 6.1 Introduction

Over the past few years, there has been a growing interest in Automatic Question Generation (AQG) - the task of generating a question from a given context (a passage, a knowledge-graph, an image, a video, *etc.*) and optionally an answer. AQG is used in creating Question Answering datasets (Li *et al.*, 2018; Sun *et al.*, 2019; Sultan *et al.*, 2020; Zhang and Bansal, 2019), enhancing user experience in conversational AI systems (Gu *et al.*, 2021; Gao *et al.*, 2019) and for creating educational material (Heilman and Smith, 2010; Kurdi *et al.*, 2020). For the above applications, it is essential that the questions are (i) grammatically correct, (ii) answerable from the passage, and (iii) specific to the given answer.

Given the practical importance of AQG, it is not surprising that there has been prolific work in this field in recent years (Jain *et al.*, 2017; Li *et al.*, 2017b; Zhang *et al.*, 2017; Du *et al.*, 2017; Duan *et al.*, 2017). However, when we started this work, there was no AQG specific evaluation metric that allowed us to compare the output of different systems. Instead, most works relied on existing  $n$ -gram based metrics such as BLEU, NIST, ROUGE, *etc.* to report their performance. However, as these metrics were not originally designed to evaluate questions, they are not adequate for capturing

the idiosyncrasies of this task (*e.g.*, they cannot evaluate whether the generated question is answerable or not). As our first step, we focus on designing a better evaluation metric for AQG which quantifies the answerability of the generated questions.

Next, we note that existing approaches based on the encode-attend-decode paradigm focus on encoding the passage, the answer, and the relationship between them using complex functions, and then generate the question in a *single pass*. However, by carefully analyzing the generated questions, we observe that these approaches tend to miss one or more essential aspects of the question. For instance, in Table 6.1, the question generated by the baseline model for the first passage is grammatically correct but is not specific to the answer. In the second example, the generated question is both syntactically incorrect and incomplete. To overcome these limitations, as our second contribution, we propose a new AQG system using the *encode-attend-refine-decode* paradigm to explicitly improve the answerability of the generated questions.

We briefly introduce the two main contributions discussed in the chapter in the following two subsections.

### 6.1.1 Q-Metric: A better metric for AQG

To begin with, we discuss a few examples below to illustrate that *answerability* depends on the presence of relevant information such as question type (Wh-types), entities, relations, *etc.* Further, we show that it is possible that a generated question has a high BLEU score but is still unanswerable and hence not useful.

Consider the task of answering questions from a Knowledge Base. Let us assume that the intended (gold standard) question is “*Who was the director of Titanic?*” and two different AQG systems generate the following questions “**S1**: *director of Titanic?*” and “**S2**: *Who was the director of?*”. Any *n*-gram based evaluation metric would assign a higher score to **S2** (BLEU3: 81.9) than **S1** (BLEU3: 36.8) as **S2** has a high overlap with the reference question. However, as should be obvious, **S1** contains all the relevant information, and most humans would be easily able to understand and answer this question. A good evaluation metric should capture this notion of *answerability* and give more importance to relevant words in the question. This brings us to the following

question, “Which are the most relevant words in a question?”

The above example might give the impression that *named entities* are essential, but other words are not. However, this is misleading and may not always be the case. For example, consider these questions, which need to be answered from an image: “Are the cats drinking milk?” versus “How many cats are drinking milk?”. These two questions have very different meanings indicating that even words like *are* and *how* are also crucial. Similarly, consider the task of answering questions from a passage titled “Matt Damon”. In this case, most humans will answer the question “What is the birthdate of” even though the named entity is missing given that the passage only talks about “Matt Damon”. Thus, in some cases, depending on the source (document, knowledge base, image), different portions of the question may be more relevant.

To concretize the intuitions developed with the help of the above examples, we first collect human judgments. Specifically, we take questions from existing datasets for document QA, knowledge base QA and visual QA and add systematic noise to these questions. We show these questions to humans and ask them to assign scores to these questions based on the *answerability* and hence the usefulness of these questions (*i.e.*, whether the question contains enough information for them to be able to answer it correctly). We also compute various *n*-gram similarity metrics (BLEU, METEOR, NIST) comparing the noisy questions to the original questions and show that these metrics do not correlate well with human judgments. Similar studies (Callison-Burch *et al.*, 2006; Liu *et al.*, 2016) have already shown that these metrics do not correlate well with *adequacy*, *coherence*, *etc.*, but here, we focus on *answerability*.

Based on the human evaluations, we propose a novel variant of existing metrics to focus on *answerability* in addition to *n*-gram similarity. The idea is to make these metrics flexible. If needed, the weight assigned to *answerability* and *n*-gram similarity can be adjusted depending on the task (document QA, Knowledge-Base QA, Visual QA). Furthermore, for capturing *answerability*, we propose additional weights for different components of the question (question type, content words, function words, and named entities). These weights can be learned from a small amount of human-annotated data and may differ from task to task.



## 6.1.2 Using Q-Metric to improve AQG systems

<b>Passage 1:</b> Liberated by <a href="#">Napoleon's</a> army in 1806, Warsaw was made the capital of the newly created Duchy of Warsaw.	
<b>Generated Questions</b>	
<i>Baseline (EAD)</i>	What was <b>the capital</b> of the newly duchy of Warsaw?
<i>EARD</i>	<b>Who liberated</b> Warsaw in 1806?
<i>Reward-EARD</i>	<b>Whose army</b> liberated Warsaw in 1806?
<b>Passage 2:</b> To fix carbon dioxide into sugar molecules in the process of photosynthesis, chloroplasts use <a href="#">an enzyme called rubisco</a>	
<b>Generated Questions</b>	
<i>Baseline (EAD)</i>	What does chloroplasts use?
<i>EARD</i>	What does chloroplasts use <b>to fix carbon dioxide into sugar molecules?</b>
<i>Reward-EARD</i>	What <b>do</b> chloroplasts use to fix carbon dioxide into sugar molecules?

Table 6.1: Samples of generated questions from Baseline (EAD), Encode-Attend-Refine-Decode (EARD), and Reward-EARD model on the SQuAD dataset. Answers are shown in [blue](#).

We saw that encode-attend-decode-based AQG systems tend to miss one or more of the essential aspects of the question in Table 6.1. The examples shown in Table 6.1 indicate that there is clear scope for improving the general quality of the questions. Additionally, the quality can be improved explicitly in terms of fluency (Example 2), answerability (Example 1), and other such aspects. One way to approach this is by revisiting the passage and answer to *refine* the initial draft by generating a better question in the second pass and then improving it for a particular aspect. We can draw a comparison between this process and how humans tend to write a rough initial draft first and then refine it over multiple passes, where the later revisions focus on improving the draft aiming at certain aspects like fluency or completeness. With this motivation, we propose a new model using the *encode-attend-refine-decode* paradigm. A small exception here is that the order of refine and decode modules is interchanged. More specifically, we first generate an initial draft and then refine it using a task-specific reward signal. The decode module generates an initial draft of the question using the passage and desired answer as inputs. The refine module then performs a second pass to generate a *refined* question by attending to both the passage and the initial draft (using a dual attention network).

From Table 6.1, we infer that our model (EARD) can generate better questions in the second pass by fixing the errors in the initial draft. We then fine-tune our model using explicit reward signals through REINFORCE with a baseline algorithm to explicitly improve the answerability (fluency) of the refined draft as compared to the initial draft. This leads to more answerable (see Reward-EARD example for first passage in Table 6.1) and fluent (see Reward-EARD example for second passage in Table 6.1) questions as compared to vanilla EARD model.

Our experiments show that the proposed model outperforms existing state-of-the-art models on the SQuAD dataset by 12.3% and 3.7% (on BLEU) given the relevant sentence and passage, respectively. We observe similar gains on the QBLEU-4 metric as well, 10.9%, and 9.91% on SQuAD sentence and passage level, respectively. We also achieve state-of-the-art results on HOTPOT-QA and DROP datasets with an improvement of 7.57% and 15.25% (on BLEU-4), 9.54% and 5.66% (on QBLEU-4) respectively over the single-decoder baseline. Our human evaluations further validate these results and show that questions generated by Reward-EARD have better fluency and answerability as compared to the questions generated by EARD. Finally, we present an interesting case study on the *originality* of the questions and show that Reward-EARD can improve the model’s performance in generating questions with high *originality*.

The remainder of this chapter is organized as follows. In Section 6.2 we explain the process of collecting human judgments for answerability. In Section 6.3 we show how we modified existing metrics to incorporate answerability. In Section 6.4, we do an extrinsic evaluation to show that the questions ranked higher using the proposed metric lead to better QA systems when used for training. In Section 6.5 we discuss our proposed model for improve both fluency and answerability of the generated questions using our *encode-attend-refine-decode* paradigm. In Section 6.7 we discuss the results and the main observations from our experiments.

## 6.2 Human Judgments For Answerability

As mentioned earlier, for AQG, in addition to  $n$ -gram similarity, we also need to focus on the *answerability* of the generated questions. As illustrated in Section 6.1.1, the

answerability of a question depends on whether it contains all relevant information, such as question type (Wh-types), named entities, and content words (often relations). Further, depending on the task (document QA, knowledge-base QA, or visual QA), the importance of these words may vary. We perform human evaluations to ascertain the importance of each of these components across different QA tasks. These evaluations allow us to independently analyze the importance of each of these components for the 3 QA tasks. In the remainder of this section, we describe (i) the process of creating noisy questions, (ii) the instructions given to the evaluators, and (iii) the inferences drawn from human evaluations.

### 6.2.1 Creating Noisy Questions

Generated Question	Reference Question
what does the ' mean ?	What is the goal of the capabilities approach ?
what is the Discovery Museum of the <b>discovery museum</b> ?	what does the Discovery Museum draw attention to ?
When was <b>Saint Victoria</b> founded ?	When was the Victoria and Albert museum founded ?
what is the <b>purpose</b> of the red phycoerytherin pigment <b>being-red</b> ?	what is the benefit of red algae being red ?

Table 6.2: Some examples to highlight that the generated questions from Baseline (EAD) models are difficult to be analyzed systematically.

One way of collecting human judgments would have been to take the output of existing AQG systems and ask humans to assign answerability scores to these questions based on the presence/absence of the above-mentioned relevant information. However, when we requested human evaluators to analyze 200 questions generated by an existing AQG system (Du *et al.*, 2017)<sup>1</sup>, they reported that the quality was often poor. We list a few sample questions used in our analysis in Table 6.2. We can see that the first two samples are not well-formed. The last two questions are well-formed, but they will lead to a very different answer as some of the keywords are incorrect or not present. In particular, after having discussions with annotators, we found that using this output would be very difficult to conduct a systematic study to assess the importance of different words in the question. Hence, we chose to use systematically simulated noisy

<sup>1</sup>We implemented the system based on the experimental setup given in this work

questions as discussed below.

We took 1000 questions each from 3 popular QA datasets, *viz.*, SQuAD, WikiMovies, and VQA. SQuAD (Rajpurkar *et al.*, 2016) is a reading comprehension dataset consisting of around 100K questions based on passages from around 500 Wikipedia articles. The WikiMovies dataset (Miller *et al.*, 2016) contains around 100K questions which can be answered from a movie knowledge graph containing 43K entities and 9 relations (*director, writer, actor, etc.*). The VQA dataset (Antol *et al.*, 2015) is an image QA dataset containing 265,016 images with around 5.4 questions on average per image. We then created noisy versions of these questions using one of the following four methods:

**Dropping function words:** We refer to the list of English function words as defined in NLTK (Loper and Bird, 2002) and drop all such words from the question. Note that a noisy question with all function words dropped will have a very low score for  $n$ -gram based metrics compared to the original question.

**Dropping Named Entities:** In our setup, identifying named entities in questions was straightforward because the questions were well-formed, and all named entities were capitalized. Alternately, we could have used the Stanford NER. However, on manual inspection, we found that marking the capitalized words as named entities were sufficient. We randomly dropped at most three named entities per question. This allows us to study how humans rate the output of an AQG system that does not contain the correct named entities.

**Dropping Content Words:** Words other than function words and named entities are also crucial for *answerability*. For example, “*Who killed Jane?*” and “*Who married Jane?*” lead to totally different answers. The word “*killed/married*” is very relevant to ascertain the correct answer. These words typically capture the relation between the entities involved (for example, *killed (John, Jane)*). We identify such important (content) words which are neither *question types* (7-Wh questions) nor *named entities* nor *stop words*. This perturbation allows us to study how humans rate an AQG system that does not produce the correct content (often relation) words.

**Changing the Question type:** Changing the question type can lead to a different

answer altogether or can make the question incoherent. For example, the answers to “Who killed Jane?” and “What killed Jane?” are entirely different. We create a noisy question by randomly changing the type of the question (for example, replace “who” with “what”). These question types are well defined (7-Wh questions including “how”), and hence it is easy to identify and replace them. This allows us to study the importance of having the correct question type in the output of an AQG system.

## 6.2.2 Instructions provided to human annotators

Rating	Description	Examples
1	All important information is missing and it is impossible to answer the question	“What is against the <del>sign</del> ?”, “Why is using <del>O2</del> instead of <del>CO2</del> less efficient?”
2	Most of the important information is missing and I can’t infer the answer to the question	“Which films did <del>Lee—H. Katzin</del> direct ?”, “Low doses of <del>anti-inflammatories</del> are sometimes used with what <del>classes</del> of drugs?”
3	Some important information is missing leading to multiple answers	“What Harvard Alumni <del>was—the</del> Palestine Prime Minister?”, “What country <del>is—the</del> teaching subject discussing?”
4	Most of the important information is present and I can infer the answer	“How <del>far</del> from the Yard is the Quad located?”, “what <del>films</del> did Melvin Van Peebles star in?”
5	All important information is present and I can answer the question	“What globally popular half marathon began <del>in</del> 1981?”, “What kind <del>of—</del> vehicle <del>is</del> parked <del>the</del> sidewalk?”

Table 6.3: Instructions along with the examples. The words that are strike out were removed from the original question to create a noisy question.

Once we generated the noisy questions, we asked human annotators to rate the *answerability* of these noisy questions on a scale of 1-5. The annotators were clearly told whether the questions were created from documents or knowledge bases, or images. In our initial evaluations, we also tried showing the actual source (image or document) to the annotators. However, we realized that this did not allow us to do an unbiased evaluation of the quality of the questions. The annotators inferred missing information from the document or image and marked the question as answerable (even though a relevant

entity was missing in the question). For example, consider the image of a cat drinking milk and the question, “*What is the drinking ?*” If a human is shown the image, she can easily infer that the missing information is “*cat*” and mark the question as answerable. This biases the study, and therefore we did not show the source to the evaluators.

A total of 25 in-house annotators participated in our study, and we got each question evaluated by 2 annotators. The annotators were Computer Science graduates competent in English. We did an initial pilot using the instructions mentioned in Table 6.3, but due to the subjective nature of the task, it was difficult for the annotators to agree on the notion of *important information*. In particular, we found that the annotators disagreed between *most important information* and *all important information* (*i.e.*, they were confused between rating 1 v/s 2 and 4 v/s 5). We, therefore, did a small pilot with a group of 10 annotators and asked them to evaluate around 30 questions from each dataset and help us refine the guidelines to define the notion of importance clearly. Based on group discussions with the annotators, we arrived at additional example based guidelines to help them distinguish between cases where “*all the*”, “*most of the*” and “*some of the*” important information is present.

The original instructions and various examples (shown in Table 6.3) were then shared and explained to all the annotators, and they used these to provide their judgments.

### 6.2.3 Human-Human Correlation

To quantify the inter-annotator agreement, we use Cohen’s kappa ( $\kappa$ ) score (Cohen, 1968) which is defined as follows:

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \quad p_e = \frac{\sum_{k=1}^5 n_{k1} \cdot n_{k2}}{N^2}$$

where  $p_o$  is similar to accuracy, *i.e.*, the ratio of the number of samples where both annotators gave the same rating to the total number of samples in the study.  $p_e$  is the probability of both the annotators agreeing by chance.  $n_{k1}, n_{k2}$  denotes the number of times annotators 1 and 2 gave a rating  $k$ , and  $N$  is the total number of samples in the study. Since each question can have five ratings,  $k$  can take on values from 1 to 5. The

$\kappa$ score	Interpretation
$< 0$	No agreement
0.0 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81 – 1.0	Perfect agreement

Table 6.4: Interpretation for the inter-annotator agreement score using Cohen’s  $\kappa$ .

Dataset	$\kappa$	Pearson	Spearman
SQuAD	0.63	0.823	0.795
WikiMovies	0.81	0.934	0.927
VQA	0.70	0.842	0.822

Table 6.5: Inter annotator agreement, Pearson and Spearman coefficients between Human Scores.

$\kappa$  score can range from  $-1$  to  $1$ . Based on the guidelines in McHugh (2012), the range of  $\kappa$  scores can be interpreted as given in Table 6.4.

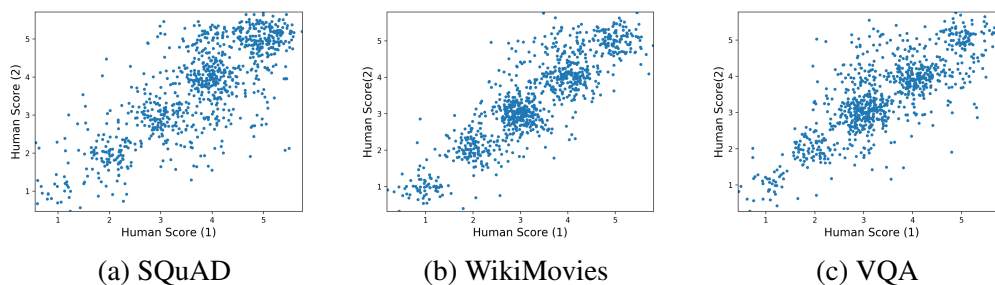


Figure 6.1: We plot the ratings given by Annotator A and B across different datasets. From the above plots, we can infer that the ratings are linearly correlated.

In Table 6.5, we report the Cohen’s kappa ( $\kappa$ ) score for rating the noisy questions created from SQuAD, WikiMovies, and VQA. We can infer from Table 6.4 and Table 6.5 that we have an almost perfect inter-annotator agreement for WikiMovies and a substantial agreement for SQuAD and VQA. Apart from reporting kappa scores, we also report the correlation between the ratings provided by the two annotators. Figure 6.1 indicates that there is a linear correlation between the two ratings for each question, and hence we measured the correlation using the Pearson coefficient. For completeness, we also measure the monotonic correlation using the Spearman coefficient. The Spearman coefficient is slightly lower than the Pearson coefficient because the inter-annotator agreement is more substantial at the tail of the distribution *i.e.* when the question is ei-

ther terrible (Rating: 1) or very good (Rating: 5).

We also observe that the correlation is good for the noisy questions created from WikiMovies, slightly lower for the questions created from VQA, and even lower for SQuAD. We investigated this further and found that in WikiMovies, it was straightforward for most annotators to figure out that named entities and relations are the most important components of the question. Hence, there was little disagreement between the ratings. However, for SQuAD, there was some bias in the judgments depending on the extent and manner in which different humans use their background knowledge for deciding whether a given question is answerable. For example, consider the question “*When did Obama Kenya?*”. One annotator felt that *Obama* and *Kenya* are the most important pieces of information. However, the question is still ambiguous, and hence it should be given a rating of 3. In contrast, another annotator felt that the missing verb (visit, criticize, leave, *etc.*) was as important as *Kenya* and *Obama* and hence gave it a rating of 1. Of course, both the evaluators agreed that the question was not answerable, but they disagreed on the extent to which it was unanswerable.

## 6.2.4 Correlation between human scores and existing evaluation metrics

Metric	SQuAD		WikiMovies		VQA	
	Pearson	Spearman	Pearson	Spearman	Pearson	Spearman
BLEU1	0.167	0.165	0.179	0.144	-0.025*	-0.048*
BLEU2	0.100*	0.103*	0.072*	0.087*	-0.075*	-0.091*
BLEU3	0.080*	0.086*	0.036*	0.001*	-0.126	-0.114
BLEU4	0.065*	0.067*	-0.020*	-0.011*	-0.086*	-0.127
ROUGE-L	0.165	0.158	0.091*	0.043*	-0.009*	-0.053*
METEOR	0.107	0.124	0.198	0.214	-0.035*	0.009*
NIST	0.173	0.158	0.088*	-0.033*	0.158	0.169

Table 6.6: Correlation between existing metrics and human judgments. The values with \* are **not** statistically significant (p-value > 0.01).

We now compare the annotator ratings to the scores assigned by various  $n$ -gram metrics, such as BLEU, METEOR, NIST, and ROUGE-L as described in Section 2.7. We first compute BLEU, METEOR, NIST, and ROUGE-L scores for each noisy question by comparing it to the original question. We then compute the correlation of each



of these scores with annotator ratings. Note that the annotator ratings are combined to obtain a gold score to compute the correlation. The ratings are normalized using the normalization method mentioned in Blatz *et al.* (2004) and then averaged to obtain the gold score.

For SQuAD and VQA, we observe that NIST, which gives more weightage to informative n-grams, correlates better than other metrics. For WikiMovies, METEOR, which even allows non-exact word matches correlates better than other metrics. For SQuAD and WikiMovies, the correlation of human scores with the simple unigram-based BLEU1 score is higher than that with other metrics. This is in line with the observation we made earlier that humans can understand and answer questions that are not well-formed, *e.g.*, “*What birth-date Damon?*”. However, overall, the correlation between human ratings and these automatic metrics is low. This suggests that it is not prudent to use these metrics for the task of AQG, given that they were originally not proposed for this task.

### 6.3 Modifying existing metrics for AQG

The above study suggests that existing metrics do not correlate well with human judgments about *answerability*. We propose modifications to these metrics so that in addition to *n*-gram similarity they also account for *answerability*. Based on the human evaluations, we found that *answerability* mainly depends on the presence of 4 types of elements, *viz.*, *relevant content words*, *named entities*, *question types* and *function words*.

As outlined in Section 6.2.1, it is easy to identify these elements in the question. Let  $c(S_r)$ ,  $c(S_n)$ ,  $c(S_q)$  and  $c(S_f)$  be the number of **r**elevant words, **n**amed entities, **q**uestion words and **f**unction words respectively in the noisy question which have corresponding matching words in the gold standard reference question. We then compute the weighted

average of the precision and recall of each of these elements as

$$P_{avg} = \sum_i w_i \frac{c(S_i)}{|l_i|} \quad (6.1)$$

$$R_{avg} = \sum_i w_i \frac{c(S_i)}{|r_i|} \quad (6.2)$$

where  $i \in \{r, n, q, f\}$ ,  $\sum_i w_i = 1$  and  $|l_i|$ ,  $|r_i|$  is the number of the words belonging to the  $i^{th}$  type of element in the noisy question and the reference question respectively. Just to be clear  $r, n, q, f$  stand for *relevant content words*, *named entities*, *question types* and *function words* respectively. Note that  $w_i$ 's are tunable weights and in Section 6.3.1, we explain how to tune these weights.

$$\text{Answerability} = 2 \cdot \frac{P_{avg} R_{avg}}{P_{avg} + R_{avg}} \quad (6.3)$$

We can combine this *answerability* score with any existing metric (say, BLEU4) to derive a modified metric for AQG as shown below:

$$Q\text{-BLEU4} = \delta \text{Answerability} + (1 - \delta) \text{BLEU4} \quad (6.4)$$

such that  $\delta \in \{0, 1\}$  to make sure that  $Q$ -Metric ranges between 0 to 1. Similarly, we can derive  $Q$ -NIST,  $Q$ -METEOR and so on.

### 6.3.1 Tuning the weights $w_i$ 's and $\delta$

We tuned the weights ( $w_i$ 's and  $\delta$ ) using the human annotation data. For each source (document, knowledge-base, and images), annotators evaluated 1000 noisy questions. The annotator scores were first scaled between 0 to 1 using the normalization method in Blatz *et al.* (2004), and the normalized scores were averaged to obtain the final gold score. We used 300 of these annotations for each source and used bagging to find the optimal weights. In particular, we drew 200 samples randomly from the given set of 300 samples and did a grid search to find  $w_i$ 's and  $\delta$  such that the  $Q$ -METRIC computed using Equation 6.4 had a maximum correlation with human scores. We repeated this process for  $k = 20$  times and computed the optimal  $w_i$ 's and  $\delta$  each time.

We found that for any given weight ( $w_i$ ), the standard deviation was very low across these  $k$  experiments. For each  $w_i$  and  $\delta$ , we obtained the final value by taking an average of the values learned across  $k$  experiments. We also observed that the weights did not change much even when we used more data for tuning. We also tuned these weights separately for each metric (*i.e.*,  $Q$ -BLEU4,  $Q$ -NIST,  $Q$ -METEOR, and so on). For illustration, we report these weights for  $Q$ -BLEU1 in Table 6.7. As expected, the weights depend on the source from which the question was generated. Specifically, for WikiMovies, named entities have the highest weight. For VQA, content words are most important, as they provide information about the entity being referred to in the question. Note that for SQuAD and VQA, the original base metric also gets weightage comparable to other components, indicating that a fluent question makes it easier to understand, thus making it answerable. The overall trend for the values of  $w_i$ 's was similar for other  $Q$ -METRICS also (*i.e.*, for  $Q$ -NIST,  $Q$ -METEOR, *etc.*).

Also, for new datasets with similar context (text/image/knowledge graph) and similar question constructions, same  $w_i$ s and  $\delta$  values can be used. However, if the dataset is significantly different from SQuAD/WikiMovies/VQA, then one could collect a small set of annotations using the same instructions, and utilize those to find new values for  $w_i$ s and  $\delta$ .

<b>n-gram metric</b>	<b>Datasets</b>	$w_{ner}$	$w_{imp}$	$w_{sw}$	$w_{qt}$	$\delta$
BLEU-1	<b>SQuAD</b>	0.41	0.36	0.03	0.20	0.66
	<b>WikiMovies</b>	0.55	0.31	0.02	0.11	0.83
	<b>VQA</b>	0.04	0.59	0.15	0.21	0.75

Table 6.7: Coefficients learnt for  $Q$ -BLEU1 from human judgments across different datasets.

### 6.3.2 Correlation between Human scores and different $Q$ -METRICS

Once the weights are tuned, we fix these weights and compute the  $Q$ -METRIC for the remaining 600-700 examples and report the correlation with human judgments for the same set of examples (see Table 6.8). For a fair comparison, the correlation scores reported in Table 6.6 are also on the same 600-700 examples. The correlation scores obtained for different  $Q$ -METRICS are indeed encouraging. In particular, we observe

Q-Metric	SQuAD		WikiMovies		VQA	
	<i>Pearson</i>	<i>Spearman</i>	<i>Pearson</i>	<i>Spearman</i>	<i>Pearson</i>	<i>Spearman</i>
<i>Q</i> -BLEU1	0.258	0.255	0.828	0.841	0.405	0.384
<i>Q</i> -BLEU2	0.244	0.243	0.825	0.835	0.390	0.360
<i>Q</i> -BLEU3	0.239	0.240	0.824	0.837	0.374	0.331
<i>Q</i> -BLEU4	0.233	0.232	0.826	0.837	0.373	0.311
<i>Q</i> -ROUGE-L	0.253	0.249	0.821	0.841	0.402	0.385
<i>Q</i> -METEOR	0.158	0.157	0.821	0.837	0.402	0.378
<i>Q</i> -NIST	0.246	0.248	0.824	0.845	0.384	0.346

Table 6.8: Correlation between proposed Q-Metric and human judgments. All the correlations have a p-value  $< 0.01$  and hence statistically significant.

that while the correlation of existing metrics with human judgments was very low (Table 6.6), the correlation of the modified metrics is much higher. This high correlation suggests that adding the learnable component for *answerability* and tuning its weights indeed leads to a better-correlated metric. Note that for VQA and SQuAD, the correlations are not as high as human-human correlations, but the correlations are still statistically significant.

### 6.3.3 Qualitative Analysis

We have listed some examples in Table 6.9, highlighting some strengths and weaknesses of *Q*-METRIC. We categorize examples as positive/negative depending on the similarity between human scores for answerability and the *Q*-BLEU score. For the examples marked as positive, the *Q*-BLEU score is very close to the *answerability* score given by humans.

## 6.4 Extrinsic evaluation

So far, we have shown that existing metrics do not always correlate well with human judgments. We have also shown that it is possible to design metrics that correlate better with human judgments by including a learnable component to focus on *answerability*. We would now like to propose an extrinsic way of evaluating the usefulness of the proposed metric. The motivation for this extrinsic evaluation comes from the fact that one of the intended purposes of the modified metrics is to use them for training QA

Dataset		Original Question	Modified Question	Human Scores	QBLEU
SQuAD	Positive	What is another type of accountant other than a CPA?	What is another type of accountant other than a ?	0.10	0.47
		In addition to schools, where else is popularly based authority effective?	In addition schools, where else popularly based authority effective?	0.85	0.83
	Negative	When did Tesla begin working for the Continental Edison Company?	When did begin working for the Continental Edison Company?	0.10	0.84
		What famous person congratulated him?	What person congratulated him?	0.85	0.17
VQA	Positive	What color is the monster truck?	What color monster truck?	0.92	0.81
		What is in the polythene ?	What is in the ?	0.10	0.14
	Negative	Why there are no leaves on the tree?	Why are leaves the tree?	0.35	0.73
		How are the carrots prepared in the plate?	How carrots prepared plate?	0.10	0.68
WikiMovies	Positive	what films does Ralf Harolde appear in ?	what films Ralf Harolde appear ?	0.97	0.91
		what is a film directed by Eddie Murphy ?	Which a film directed by Eddie Murphy ?	0.91	0.88
	Negative	what films does Gerard Butler appear in ?	how does Gerard Butler appear in ?	0.15	0.89
		John Conor Brooke appears in which movies ?	appears in which movies ?	0.03	0.44

Table 6.9: Human (Gold) and  $Q$ -Metric scores for some of the examples from the collected human-evaluation data.

systems. Suppose we use a particular metric for evaluating the quality of an AQG system and suppose this metric suggests that the questions generated by this system are poor. We would obviously discard this system and not use the questions generated by it to train a QA system. However, if the metric itself is questionable, then it is possible that the questions were good enough, but the metric was not good to evaluate their quality.

To study this effect, we create a noisy version of the training data of SQuAD, WikiMovies, and VQA using the same methods outlined in Section 6.2.1. We then train a state-of-the-art model for each of these tasks on this noisy data and evaluate the trained model on the original test set of each of these datasets. The models that we considered were Seo *et al.* (2016) for SQuAD, Miller *et al.* (2016) for WikiMovies and Ben-younes *et al.* (2017) for VQA.

The results of our experiments are summarized in Tables 6.10 - 6.12. The first column for each table shows how the noisy training data was created. The second column shows the BLEU4 score of the noisy questions compared to the original reference questions (thus, it tells us the perceived quality of these questions under the BLEU4 metric).

Type of Noise	BLEU	QBLEU	Hit 1
None	100	100	76.5
Stop Words	25.4	84.0	75.6
Question Type	74.0	79.3	73.5
Content Words	29.4	64.3	54.7
Named Entity	41.9	48.5	17.97

Table 6.10: Performance obtained by training on different types of noisy questions (WikiMovies).

Noise	BLEU	QBLEU	F1
None	100	100	76.5
Question Type	80.1	66.1	69.0
Stop Words	24.2	61.0	70.4
Content Words	60.7	57.1	64.1
Named Entity	77.0	56.0	73.8

Table 6.11: Performance obtained by training on different types of noisy questions (SQuAD).

Noise	BLEU	QBLEU	Acc(%)
None	100	100	64.4
Content Words	49.4	58.2	60.21
Question Type	63.7	50.9	59.81
Stop Words	10.8	37.7	57.37

Table 6.12: Performance obtained by training on different types of noisy questions (VQA).

Similarly, the third column tells us the perceived quality of these questions under the  $Q$ -BLEU4 metric. Ideally, we would want that the model’s performance should correlate better with the perceived quality of the training questions as identified by a given metric. We observe that the general trend is better *w.r.t.* the  $Q$ -BLEU4 metric than the BLEU4 metric (*i.e.*, in general, higher  $Q$ -BLEU4 indicates better performance, and lower  $Q$ -BLEU4 indicates poor performance). In particular, we notice that BLEU4 gives much importance to stop words, but these words hardly influence the final performance of the QA system. We believe that such an extrinsic evaluation should also be used while designing better metrics to help get better insights.

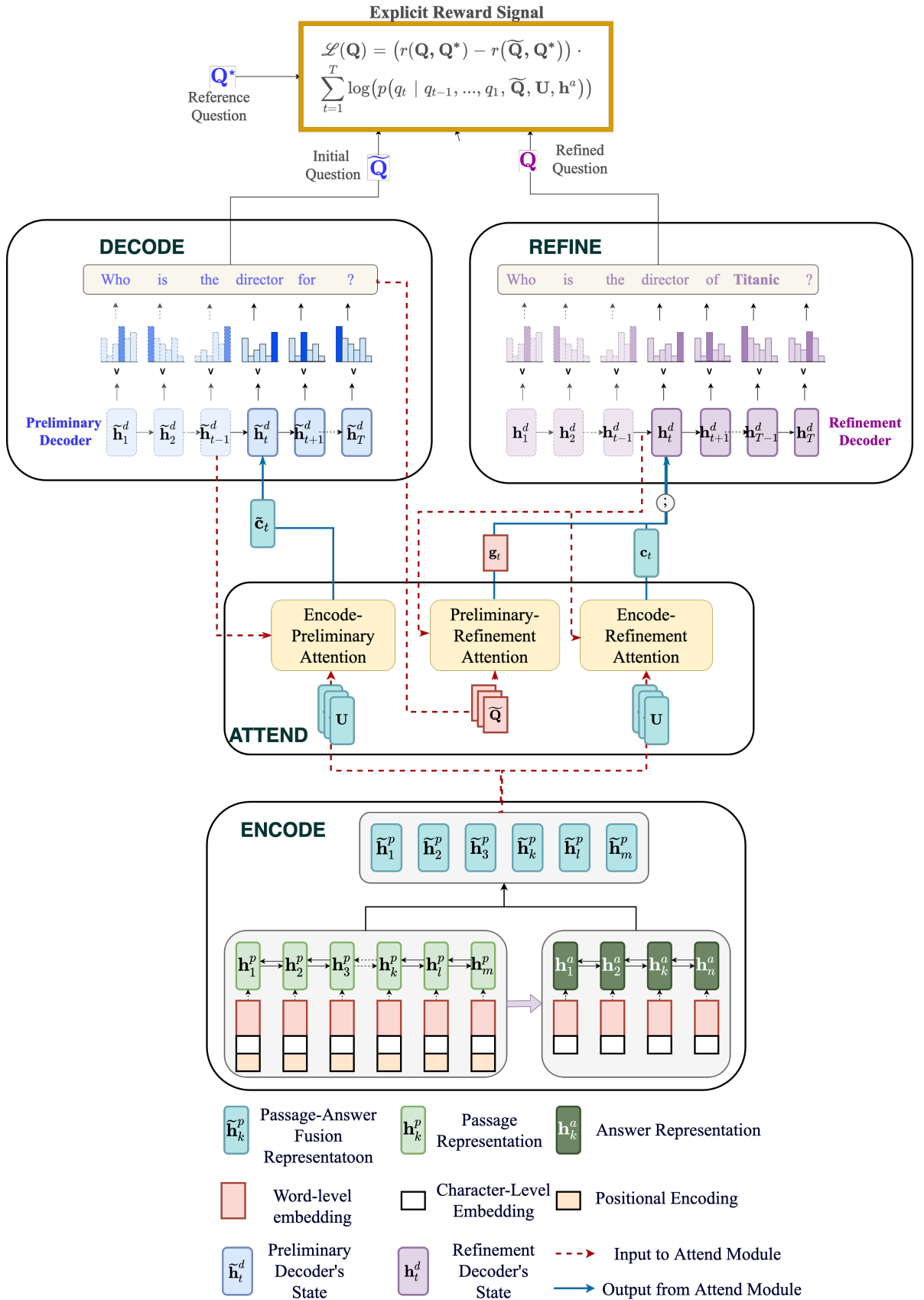


Figure 6.2: Our Proposed Model with encode-attend-refine-decode paradigm. The de-code module provides an initial draft of the question to the refine module. The refine module then improves this initial draft by attending to the initial draft and the passage-answer representation simultaneously.

## 6.5 Encode-Attend-Refine-Decode Model for AQQ

Having designed a metric that captures task-specific characteristics, we now propose a model based on our *encode-attend-refine-decode* paradigm, which also uses this metric to improve the answerability of the generated questions. As mentioned earlier, a small exception here is that the order of the refine and decode modules is interchanged. More specifically, we first generate an initial draft and then refine it using task-specific reward signals. One could also see this as *encode-attend-decode-refine-decode* where the refinement happens due to (i) the extra information that the second decoder receives from the preliminary decoder and (ii) the reward signal obtained using the proposed *Q*-Metric.

For a given passage  $\mathbf{P} = \{w_1^p, \dots, w_m^p\}$  of length  $m$  and answer  $\mathbf{A} = \{w_1^a, \dots, w_n^a\}$  of length  $n$ , we first obtain *answer-aware* latent representation,  $\mathbf{U} = \{\tilde{\mathbf{h}}_1^p, \dots, \tilde{\mathbf{h}}_m^p\}$ , for every word of the passage and an answer representation  $\mathbf{h}^a$  (as described in Section 6.5.1). We then generate an initial draft  $\tilde{\mathbf{Q}} = \{\tilde{q}_1, \dots, \tilde{q}_T\}$  by computing  $\tilde{q}_t$  as

$$\tilde{q}_t = \arg \max_{\tilde{q}} \prod_{t=1}^l \tilde{p}(\tilde{q}_t | \tilde{q}_{t-1}, \dots, \tilde{q}_1, \mathbf{U}, \mathbf{h}^a) \quad (6.5)$$

Here  $\tilde{p}(\cdot)$  is a probability distribution modeled using the Preliminary Decoder. We then refine the initial draft  $\tilde{\mathbf{Q}}$  using the Refinement Decoder to obtain the refined draft  $\mathbf{Q} = \{q_1, \dots, q_T\}$ :

$$q_t = \arg \max_q \prod_{t=1}^l p(q_t | q_{t-1}, \dots, q_1, \tilde{\mathbf{Q}}, \mathbf{U}, \mathbf{h}^a) \quad (6.6)$$

We then use explicit rewards to enforce refinement on the desired metric, such as answerability and fluency. In the subsequent subsections, we will describe the different components of our model and our reward mechanism.

### 6.5.1 Encode Module

Our encode module consists of three layers: (i) Embedding, (ii) Contextual, and (iii) Passage-Answer Fusion layers. To capture the interaction between passage and answer,



we ensure that the passage and answer representations are fused at every layer.

**Embedding Layer:** We compute a  $d$ -dimensional embedding for every word in the passage and the answer in this layer. This embedding is obtained by concatenating the word’s Glove embedding (Pennington *et al.*, 2014) with its character-level embedding as discussed in Seo *et al.* (2016). Additionally, for passage words, we also compute a positional embedding based on the relative position of the word *w.r.t.* the answer span as described in Zhao *et al.* (2018). For every passage word, this positional embedding is also concatenated to the word-level and character-level embeddings. For answer words, we use only word-level and character-level embeddings. We discuss the impact of character embeddings and answer tagging in later sections. In the subsequent sections, we will refer to the embedding of the  $i$ -th passage word  $w_i^p$  as  $e(\mathbf{w}_i^p)$  and the  $j$ -th answer word  $w_j^a$  as  $e(\mathbf{w}_j^a)$ .

**Contextual Layer:** In this layer, we compute a contextualized representation for every word in the passage by passing the word embeddings (as computed above) through a bidirectional-LSTM network:

$$\vec{\mathbf{h}}_t^p = \text{LSTM}(e(\mathbf{w}_t^p), \vec{\mathbf{h}}_{t-1}^p) \quad \forall t \in [1, m]$$

where  $\vec{\mathbf{h}}_t^p$  is the hidden state of the forward LSTM at time  $t$ . We then concatenate the forward and backward hidden states as  $\mathbf{h}_t^p = [\vec{\mathbf{h}}_t^p; \overleftarrow{\mathbf{h}}_t^p]$ .

The answer could correspond to a span in the passage. Let  $j + 1$  and  $j + n$  be the start and end indices of the answer span in the passage respectively. We can thus refer to  $\{\mathbf{h}_{j+1}^p, \dots, \mathbf{h}_{j+n}^p\}$  as the representation of the answer words in the context of the passage. We then obtain contextualized representations for the  $n$  answer words by passing them through an LSTM as follows:

$$\vec{\mathbf{h}}_t^a = \text{LSTM}([e(\mathbf{w}_t^a), \mathbf{h}_{j+t}^p], \vec{\mathbf{h}}_{t-1}^a) \quad \forall t \in [1, n]$$

**Passage-Answer Fusion Layer:** In this layer, we refine the representations of the pas-

sage words based on the answer representation as follows:

$$\tilde{\mathbf{h}}_i^p = \tanh(\mathbf{W}_u [\mathbf{h}_i^p; \mathbf{h}^a; \mathbf{h}_i^p \odot \mathbf{h}^a]) \quad \forall i \in [1, m]$$

Here  $\mathbf{W}_u \in \mathbb{R}^{l \times 3l}$  and  $l$  is the size of the hidden state of the LSTM. This is similar to how Seo *et al.* (2016) capture interactions between passage and question for QA. We use  $\mathbf{U} = \{\tilde{\mathbf{h}}_1^p, \dots, \tilde{\mathbf{h}}_m^p\}$  as the fused passage-answer representation which is then used by our decoder(s) to generate the question  $\mathbf{Q}$ .

## 6.5.2 Attend Module

The attend module typically assists the decoder select relevant context from passage/input to generate the next word. In this chapter, the generation of questions happens in two passes: a Preliminary Decoder generates an initial draft followed by a Refinement Decoder that generates a revised draft. For the revised draft, the Refinement Decoder should attend to both the initial draft and the passage. If the Refinement Decoder attends to only one of them, it may fail to improve upon the initial draft. To this end, our attend module consists of three sub-components as listed below.

**Encoder-to-PreliminaryDecoder attention network (E2P):** This network computes an attention weighted sum of the contextualized passage word representations  $\mathbf{U}$ , which is then passed to the Preliminary Decoder to predict the next word at timestep  $t$ . This representation is computed as follows:

$$\tilde{\mathbf{c}}_t = \sum_{i=1}^m \alpha_t^i \tilde{\mathbf{h}}_i^p \quad (6.7)$$

where  $\alpha_t^i$  is a parameterized and normalized attention weight computed as:

$$a_{t,i} = \mathbf{v}_p^T \cdot \tanh(\mathbf{W}_p \cdot \tilde{\mathbf{h}}_t^d + \mathbf{V}_p \cdot \tilde{\mathbf{h}}_i^p) \quad (6.8)$$

$$\alpha_{t,i} = \frac{\exp(a_{t,i})}{\sum_{j=1}^n \exp(a_{t,j})} \quad (6.9)$$

where  $\tilde{\mathbf{h}}_t^d$  is the hidden state of the Preliminary Decoder's LSTM at timestep  $t$ , and  $\mathbf{v}_p, \mathbf{W}_p, \mathbf{V}_p$  are network parameters.

**PreliminaryDecoder-to-RefinementDecoder attention network (P2R):** This network computes an attention weighted sum of the embeddings  $\mathbf{g}_t$  of the words generated by the Preliminary Decoder, which is then passed to the Refinement Decoder to predict the next word at timestep  $t$ .  $\mathbf{g}_t$  is computed as follows:

$$\mathbf{g}_t = \sum_{i=1}^T \beta_i^t \mathbf{e}_w(\tilde{\mathbf{q}}_i) \quad (6.10)$$

where  $\beta_i^t$  is a parameterized and normalized attention weight computed as:

$$b_{t,i} = \mathbf{v}_r^T \cdot \tanh(\mathbf{W}_r \cdot \mathbf{h}_t^d + \mathbf{V}_r \cdot \mathbf{e}_w(\tilde{q}_i)) \quad (6.11)$$

$$\beta_{t,i} = \frac{\exp(b_{t,i})}{\sum_{j=1}^n \exp(b_{t,j})} \quad (6.12)$$

where  $\mathbf{h}_t^d$  is the hidden state of the Refinement Decoder at timestep  $t$ ,  $\{\mathbf{v}_r, \mathbf{W}_r, \mathbf{V}_r\}$  are network parameters and  $\mathbf{e}_w(\tilde{q}_i)$  refers to the word embedding of the word  $\tilde{q}_i$  generated in the initial draft.

**Encoder-to-RefinementDecoder attention network (E2R):** Since the initial draft could be erroneous or incomplete, we obtain additional information from the passage instead of only relying on the output of the Preliminary Decoder. We do so by computing a context vector  $\mathbf{c}_t$ , which is then passed to the Refinement Decoder. It is computed as follows:

$$\mathbf{c}_t = \sum_{i=1}^m \gamma_i^t \tilde{\mathbf{h}}_i^p \quad (6.13)$$

where  $\gamma_i^t$  is computed as:

$$d_{t,i} = \mathbf{v}_d^T \cdot \tanh(\mathbf{W}_d \cdot \mathbf{h}_t^d + \mathbf{V}_d \cdot \tilde{\mathbf{h}}_i^p) \quad (6.14)$$

$$\gamma_{t,i} = \frac{\exp(d_{t,i})}{\sum_{j=1}^n \exp(d_{t,j})} \quad (6.15)$$

where  $\mathbf{h}_t^d$  is the hidden state of the Refinement Decoder at timestep  $t$ ,  $\{\mathbf{v}_d, \mathbf{W}_d, \mathbf{V}_d\}$  are network parameters.

### 6.5.3 Decode Module

The Preliminary Decoder generates an initial draft, one word at a time using an LSTM network as follows:

$$\tilde{\mathbf{h}}_t^d = \text{LSTM}([\mathbf{e}_w(\tilde{\mathbf{q}}_{t-1}); \tilde{\mathbf{c}}_{t-1}; \mathbf{h}^a], \tilde{\mathbf{h}}_{t-1}^d) \quad (6.16)$$

where  $\tilde{\mathbf{h}}_t^d$  is the hidden state of the LSTM at timestep  $t$ ,  $\mathbf{h}^a$  is the answer representation as computed above,  $\tilde{\mathbf{c}}_{t-1}$  is context vector obtained through the E2P attention network as given in Equation 6.7.  $\mathbf{e}_w(\tilde{\mathbf{q}}_t)$  is the embedding of the word  $\tilde{q}_t$ .

We estimate the probability distribution as defined in Equation (6.5), over the output vocabulary to obtain  $\tilde{q}_t$  at timestep  $t$  as follows:

$$\tilde{p}(\tilde{q}_t) = \text{softmax}(\mathbf{W}_o \cdot [\mathbf{W}_c \cdot [\tilde{\mathbf{h}}_t^d; \tilde{\mathbf{c}}_t]]), \quad (6.17)$$

where  $\mathbf{W}_c$  is network parameter and  $\mathbf{W}_o$  is the output matrix which projects the final representation to  $\mathbb{R}^V$  where  $V$  is the vocabulary size.  $\tilde{q}_t$  is the word which has the highest probability under the above estimated distribution.

### 6.5.4 Refine Module

Once the Preliminary Decoder generates the entire question, the Refinement Decoder uses it to generate an updated version of the question. The hidden state of the Refinement Decoder at time  $t$  is computed as follows:

$$\mathbf{h}_t^d = \text{LSTM}([\mathbf{e}_w(\mathbf{q}_{t-1}); \mathbf{c}_{t-1}; \mathbf{g}_{t-1}; \mathbf{h}^a], \mathbf{h}_{t-1}^d) \quad (6.18)$$

We estimate the probability distribution as defined in Equation (6.6), over the output vocabulary to obtain  $q_t$  at timestep  $t$  as follows:

$$p(q_t) = \text{softmax}(\mathbf{W}_o \cdot [\mathbf{W}'_c \cdot [\mathbf{h}_t^d; \mathbf{c}_t; \mathbf{g}_t]]) \quad (6.19)$$

where  $\mathbf{W}'_c$  is a network parameter and  $\mathbf{W}_o$  is the output matrix which is shared with the Preliminary Decoder (Equation 6.17).  $q_t$  is the word which has the highest probability under the above estimated distribution. Note that our model generates two variants of the question : initial draft  $\tilde{\mathbf{Q}}$  and refined draft  $\mathbf{Q}$ . We compare these two versions of the generated questions in Section 6.7.

### 6.5.5 Training Objective

The objective function commonly used by the decoder is the negative log-likelihood of the generated sequence. As we saw in Table 6.1, this does not necessarily ensure that the generated question is better in terms of *answerability* or *fluency*.

The primary goal of our proposed model of generating two drafts is to improve the second draft in terms of desired qualities of a question, such as : answerability and fluency as compared to the initial draft. To this end, we use “REINFORCE with a baseline” algorithm (Williams, 1992). We first compute the reward  $r(\tilde{\mathbf{Q}}, \mathbf{Q}^*)$  and  $r(\mathbf{Q}, \mathbf{Q}^*)$  for the question generated by the Preliminary and Refinement Decoders respectively. We reward Refinement Decoder using the Preliminary Decoder’s reward  $r(\tilde{\mathbf{Q}}, \mathbf{Q}^*)$  as the baseline. More specifically, given the Preliminary Decoder’s generated word sequence  $\tilde{\mathbf{Q}} = \{\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_T\}$  and Refinement Decoder’s generated word sequence  $\mathbf{Q} = \{q_1, q_2, \dots, q_T\}$  obtained from the distribution  $p(q_t | q_{t-1}, \dots, q_1, \tilde{\mathbf{Q}}, \mathbf{U}, \mathbf{h}^a)$ , the training loss is defined as follows

$$L(\mathbf{Q}) = (r(\mathbf{Q}, \mathbf{Q}^*) - r(\tilde{\mathbf{Q}}, \mathbf{Q}^*)) \cdot \sum_{t=1}^T \log p(q_t | q_{t-1}, \dots, q_1, \tilde{\mathbf{Q}}, \mathbf{U}, \mathbf{h}^a)$$

where  $r(\mathbf{Q})$  and  $r(\tilde{\mathbf{Q}})$  are the rewards obtained by comparing with the reference question  $\mathbf{Q}^*$ . As mentioned, this reward  $r(\cdot)$  is the answerability score as defined in Section 6.1.1. We can even focus on increasing the fluency of the refined draft by using BLEU as the reward function.

Note that to handle out of the vocabulary words in generated questions, we adopt the same strategy as given in Section 5.2.5.

## 6.6 Experimental Details

This section gives a brief description of the datasets used to compare our models with baseline models. We also provide implementation details of for our model as well as the baselines.

### 6.6.1 Datasets for AQG

**SQuAD** (Rajpurkar *et al.*, 2016): As described earlier, it contains 100K (question, answer) pairs obtained from 536 Wikipedia articles, where the answers are a span in the passage. For SQuAD, AQG has been tried from both sentences and passages. In the former case, only the sentence which contains the answer span is used as input, whereas in the latter case, the entire passage is passed as an input. We use the same train-valid-test splits as used in Zhao *et al.* (2018).

**Hotpot QA** (Yang *et al.*, 2018) : Hotpot-QA is a multi-document and multi-hop QA dataset. Along with the triplet (passage, answer, question), the authors also provide supporting facts that potentially lead to the answer. We concatenate these supporting facts to form the passage. The answers here are either yes/no or an answer span in passage. We use 10% of the training data for validation and use the original dev set as test set. This average length of the questions in this dataset is around 17 words, which is relatively higher than the other two datasets.

**DROP** (Dua *et al.*, 2019): This is a reading comprehension benchmark that requires discrete reasoning over the passage. It contains 96K questions which require discrete operations such as addition, counting, or sorting to obtain the answer. We use 10% of the original training data for validation and use the original dev set as the test set.

### 6.6.2 Implementation Details

We use 300 dimensional pre-trained Glove word embeddings, which are fixed during training. For character-level embeddings, we initially use a 20 dimensional embedding for the characters, which is then projected to 100 dimensions. For answer-tagging, we

use an embedding size of 3. The hidden size for all the LSTMs is fixed to 512. We use 2-layer, 1-layer, 2-layer, and 2-layer stacked BiLSTM in passage-level RNN, answer-level RNN, Preliminary Decoder’s RNN, and Refinement Decoder’s RNN, respectively. We take the top 30,000 frequent words as the vocabulary. We use Adam optimizer with a learning rate of 0.0004 and train our models for 10 epochs using cross-entropy loss. For the Reward-**EARD** model, we fine-tune the pre-trained model with the loss function mentioned in Section 6.5.5 for 3 epochs. The best model is chosen based on the BLEU (Papineni *et al.*, 2002) score on the validation split. For all the results, we use beam search decoding with a beam size of 5.

## 6.7 Results and Discussions

This section presents the results and analysis of our proposed model, which we refer to as **EARD** (Encode-Attend-Refine-Decode). We compare our model to the vanilla Encode-Attend-Decode model that we refer to as **EAD**. Note that the performance of this model is comparable to our implementation of the model proposed in Zhao *et al.* (2018). We evaluate our models based on  $n$ -gram similarity metrics BLEU, ROUGE-L, and METEOR. We also quantify the answerability of our models using our proposed metric, QBLEU-4.

In this section, we (i) compare **EARD**’s performance with **EAD** and existing models across all the datasets mentioned above (ii) report human evaluations to compare **EARD** and **EAD** (iii) analyze initial and refined draft generated from Preliminary Decoder and Refinement Decoder respectively, and (iv) present the performance of Reward-**EARD** with two different reward signals (*fluency* and *answerability*).

### 6.7.1 **EARD**’s performance across datasets

In Table 6.13, we compare the performance of **EARD** with existing models and **EAD** across different datasets. We also report the current state of the art result for SQuAD sentence level dataset which uses a transformer-based sequence-to-sequence model Xiao *et al.* (2020). Note that for a fair comparison, we train our **EARD** model us-

Model	n-gram						QBLEU4
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	
<b>SQuAD (Sentence Level)</b>							
Sun <i>et al.</i> (2018)	43.02	28.14	20.51	15.64	-	-	-
Zhao <i>et al.</i> (2018)	44.51	29.07	21.06	15.82	44.24	19.67	-
Kim <i>et al.</i> (2018)	-	-	-	16.17	-	-	-
EAD	44.74	29.79	22.00	16.84	44.78	20.60	24.7
EARD	<b>47.27</b>	<b>31.88</b>	<b>23.65</b>	<b>18.16</b>	<b>47.14</b>	<b>23.40</b>	<b>27.4</b>
SOTAXiao <i>et al.</i> (2020)	-	-	-	25.40	26.92	52.84	-
<b>SQuAD (Passage Level)</b>							
Zhao <i>et al.</i> (2018)	45.07	29.58	21.60	16.38	44.48	20.25	-
EAD	44.61	29.37	21.50	16.36	43.95	20.11	24.2
EARD	<b>46.41</b>	<b>30.66</b>	<b>22.42</b>	<b>16.99</b>	<b>45.03</b>	<b>21.10</b>	<b>26.6</b>
<b>HOTPOT-QA</b>							
Zhao <i>et al.</i> (2018)*	45.29	32.06	24.43	19.29	40.40	19.29	25.7
EAD	<b>46.00</b>	32.47	24.82	19.68	41.52	23.27	26.2
EARD	45.45	<b>33.13</b>	<b>26.05</b>	<b>21.17</b>	<b>43.12</b>	<b>25.81</b>	<b>28.7</b>
<b>DROP</b>							
Zhao <i>et al.</i> (2018)*	39.56	29.19	22.53	18.07	45.01	19.68	31.4
EAD	39.21	29.10	22.65	18.42	45.07	19.56	31.8
EARD	<b>42.81</b>	<b>32.63</b>	<b>25.78</b>	<b>21.23</b>	<b>47.49</b>	<b>22.25</b>	<b>33.6</b>

Table 6.13: Comparison of EARD model with existing approaches and EAD model. Here \* denotes our implementation of the corresponding work.

ing only cross-entropy loss, and in later sections we present the results obtained using the training objective described in Section 6.5.5. On BLEU-4 metric, **EARD** beats the existing state-of-the-art model by 12.30%, 9.74%, 17.48%, and 3.71% respectively on SQuAD (sentence), HOTPOT-QA, DROP and SQuAD (passage) dataset. Also it outperforms **EAD** by 7.83%, 7.57%, 15.25% and 3.85% respectively on SQuAD (sentence), HOTPOT-QA, DROP and SQuAD (passage). In general, **EARD** is consistently better than existing models across all  $n$ -gram scores (BLEU, ROUGE-L, and METEOR). Along with  $n$ -gram scores, we observe improvements on Q-BLEU4 as well, which, as described earlier, gives a measure of both answerability and fluency.

## 6.7.2 Human Evaluations

We conducted human evaluations to analyze the quality of the questions produced by **EAD** and **EARD**. We randomly sampled 500 questions generated from the SQuAD (sentence level) dataset and asked the annotators to compare the quality of the generated questions. The annotators were shown a pair of questions, one generated by **EAD** and a revised draft generated by **EARD** from the same sentence, and were asked to choose which one was better in terms of *fluency*, *completeness*, and *answerability*. Note that



---

**Passage:** Before the freeze ended in 1952, there were only 108 existing television stations in the United States; a few major cities (such as Boston) had only **two** television stations, ...

---

**Questions**

**EAD:** how many television stations existed in boston ?

**EARD:** how many television stations did boston have in the united ?

---

Table 6.14: An example where **EAD** model was better than **EARD**. The ground truth answers are shown in **blue**.

Model	Decoder	BLEU-4	QBLEU-4
<b>EAD</b>	-	16.84	24.70
without <b>P2R</b> attention	<i>Refinement Decoder</i>	17.16	25.80
	<i>Preliminary Decoder</i>	17.59	26.00
with <b>P2R</b> attention	<i>Refinement Decoder</i>	18.37	27.40
	<i>Preliminary Decoder</i>	17.89	26.00

Table 6.15: Comparison between Preliminary and Refinement decoders in **EARD** Model for SQuAD Sentence Level QG.

this required the annotators to annotate in a comparative setting rather than giving an absolute score. They were allowed to skip the question pairs where they could not make a clear choice. Three annotators rated each question, and the final label was calculated based on majority voting. We observed that the **EARD** model outperforms the **EAD** model across all three metrics. Over 68.6%, 66.7% and 64.2% of the generated questions from **EARD** were respectively more fluent, complete and answerable when compared to the **EAD** model. However, there are some cases where **EAD** does better than **EARD**. For example, in Table 6.14, we show that while trying to generate a more elaborate question, **EARD** introduces an additional phrase “*in the united*” which is not required. Due to such instances, annotators preferred the **EAD** model in around 30% of the instances.

### 6.7.3 Analysis of Refined Draft and Initial Draft

The Preliminary and Refinement Decoders impact each other through two paths: (i) indirect path, where they share the encoder and the output projection to the vocabulary  $V$ , (ii) direct path, via the **P2R** attention network, where the Refinement Decoder attends to the initial draft of the question. When **EARD** has only an indirect path, we can infer from rows 1 and 3 of Table 6.15 that the performance of Preliminary Decoder improves

<i>Sample</i>	<p><b>Sentence:</b> The antigens expressed by tumors have several sources ; some are derived from oncogenic viruses like <a href="#">human papillomavirus</a> , which causes cervical cancer , while others are the organism ’s own proteins that occur at low levels in normal cells but reach high levels in tumor cells</p> <p><b>Reference Question:</b> What is the virus in humans that causes cervical cancer ?</p>
<i>Generated Questions</i>	<p><b>Refined Draft:</b> What is the name of the virus that causes cervical cancer?</p> <p><b>Initial Draft:</b> What is the name of the oncogenic virus?</p>
<b>Sample</b>	<p><b>Sentence:</b> Tesla considered the winter of <a href="#">1886/1887</a> as a time of “ terrible headaches and bitter tears . ”</p> <p><b>Reference:</b> which years did tesla refer to as a time of terrible headaches and bitter tears ?</p>
<i>Generated Questions</i>	<p><b>Refined Draft:</b> what year did tesla consider the winter of “ terrible headaches and bitter tears ” ?</p> <p><b>Initial Draft:</b> what year did tesla consider the winter of bitter tears ?</p>

Table 6.16: Generated samples by Preliminary and Refinement Decoders in our proposed **EARD** model. We observe that the Refinement Decoder improves the draft by adding relevant phrases “*{causes cervical cancer, terrible headaches}*”.

when compared to the EAD model (16.84 v/s 17.59 BLEU). This suggests that generating two variants of the question improves the performance of the initial draft as well. This is perhaps due to the additional feedback that the shared encoder and output layer get from the Refinement Decoder. When we add the direct path (**P2R** attention network) between the two decoders, the performance of the Refinement Decoder, as well as the Preliminary Decoder, improves as shown in rows 4 and 5 of the Table 6.15.

**Comparison on Answerability:** We also evaluate both the initial and refined draft using QBLEU4. We observe that the increase in Q-Metric for refined questions is because the **EARD** model can correct/add the relevant Named Entities in the question. In particular, we observe that the Named Entity component score in Q-Metric increases from 32.42 for the first draft to 37.81 for the refined draft.

**Qualitative Analysis:** Figure 6.3 shows that the Refinement Decoder indeed generates more detailed questions when compared to the initial draft generated from the Preliminary Decoder. As shown in Table 6.16, the quality of the refined questions is better than the initial drafts of the questions. For Example 1, Refinement Decoder adds the phrase “*causes cervical cancer*” to make the question more specific to the answer. In the case of Example 2, Refinement Decoder makes the question more complete by adding phrase

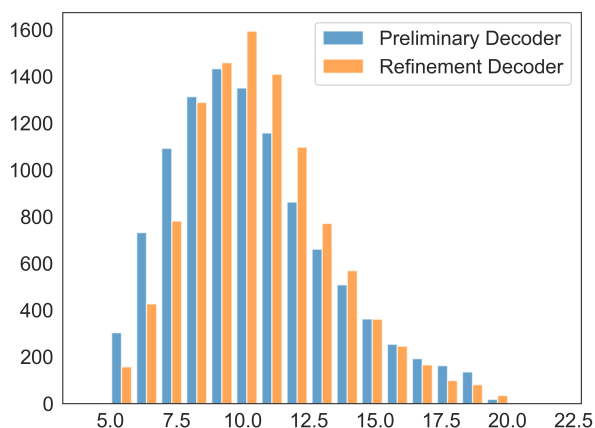


Figure 6.3: Generated Question Length Distribution for Preliminary and Refinement Decoders.

*“terrible headaches”*

Apart from adding phrases to make the initial draft more specific or complete, the Refinement Decoder is also able to make certain corrections in the initial draft. For instance in 6.17, we have enumerated instances where the Refinement Decoder corrects the question types and function words in the refined draft.

**Visualization of Attention plots from all three attention networks** We plot the aggregated attention given to the passage and the initial draft of the generation question across the Refinement Decoder’s various time-steps in Figure 6.4 for Example 1 in Table 6.16. Although both the questions are specific to the answer, the initial question is very generic. **E2R** attention network pays attention to the context surrounding the answer, which leads to a complete question by adding the phrase *“causes cervical cancer”*. Also, note that in the **P2R** attention network, while attending to the initial draft, the word *“oncogenic”* is not paid attention to, which allows the Refinement Decoder to generate a better question.

#### 6.7.4 Analysis of Reward Based Training Objective

In this section, we analyze the impact of employing different reward signals in **EARD**. As discussed earlier in section 6.5.5, we use *answerability* and *fluency* scores as reward signals. As shown in Table 6.18, when *answerability* score is used as a reward signal, there is an improvement in QBLEU-4 scores of Reward-**EARD** as compared to the

<i>Sample</i>	<b>Sentence:</b> Any member can put their name forward to be First Minister , and a vote is taken by all members of Parliament . <b>Reference Question:</b> Who is eligible to toss their name in the hat to be First Minister ?
<i>Generated Questions</i>	<b>Refined Draft:</b> who can put their name to be first minister ? <b>Initial Draft:</b> what can put their name forward to be first minister ?
<b>Sample</b>	<b>Sentence:</b> Downtown San Diego is the central business district of San Diego , though the city is filled with business districts . <b>Reference:</b> What is the central business district of San Diego ?
<i>Generated Questions</i>	<b>Refined Draft:</b> what is the central business district of san diego ? <b>Initial Draft:</b> how is the central business district of san diego located ?
<b>Sample</b>	<b>Sentence:</b> The further decline of Byzantine state-of-affairs paved the road to a third attack in 1185 , when a large Norman army invaded Dyrrachium , owing to the betrayal of high Byzantine officials . <b>Reference:</b> When did the Normans attack Dyrrachium ?
<i>Generated Questions</i>	<b>Refined Draft:</b> in what year did the norman army attack Dyrrachium ? <b>Initial Draft:</b> in what year did a norman army attack Dyrrachium ?

Table 6.17: Generated samples by Preliminary and Refinement Decoders in our proposed **EARD** model. We observe that the Refinement Decoder improves the draft by correcting question types/function words in the initial draft.

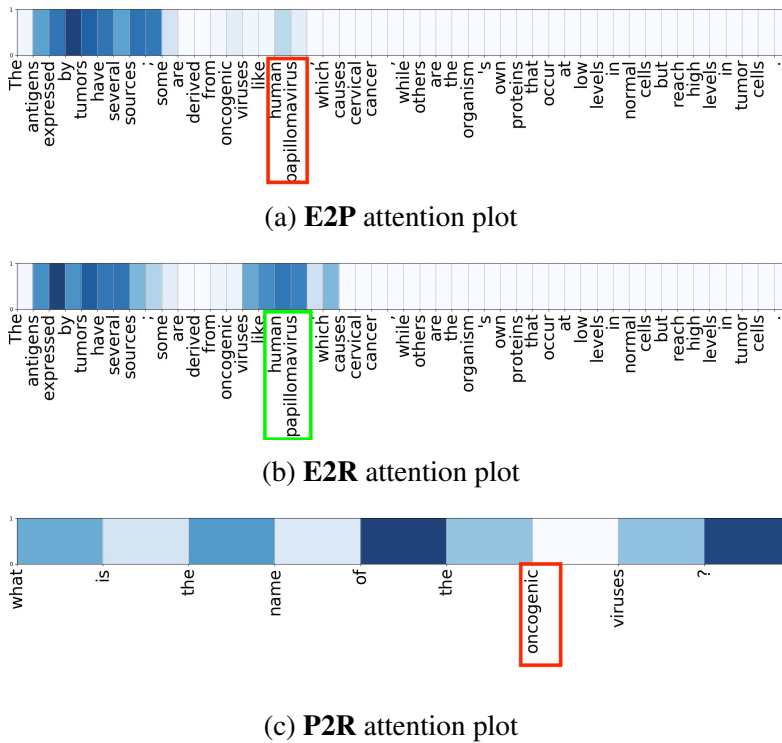


Figure 6.4: Attention plots for a) E2P, b) E2R, c) P2R respectively

**Initial Draft:** “What is the name of the oncogenic virus?”

**Refined Draft:** “What is the name of the virus that causes cervical cancer?”

Datasets	Model	BLEU-4 Reward Signal	Answerability Reward Signal
SQuAD (Sentence Level)	<b>EARD</b>	18.37	36.9
	Reward- <b>EARD</b>	18.52	37.5
SQuAD (Passage Level)	<b>EARD</b>	16.99	26.6
	Reward- <b>EARD</b>	17.11	27.3
HOTPOT-QA	<b>EARD</b>	21.17	28.7
	Reward- <b>EARD</b>	21.32	29.2
DROP	<b>EARD</b>	21.23	33.6
	Reward- <b>EARD</b>	21.60	34.3

Table 6.18: Impact of Reward-**EARD** on various datasets when fluency and answerability are used as reward signals.

---

**Passage:** *Cost engineers and estimators* apply expertise to relate the work and materials involved to a proper valuation

---

**Questions**

*Generated:* Who apply expertise to relate the work and materials involved to a proper valuation ?

*True:* Who applies expertise to relate the work and materials involved to a proper valuation ?

---

Table 6.19: An example of question with significant overlap with the passage. The answer is shown in *blue*.

**EARD** model across all datasets. We validated these results through human evaluations across 200 samples for SQuAD (sentence level) dataset. Annotators prefer the Reward-**EARD** model in 70% of the cases for answerability. Similarly, when we use the BLEU-4 score as a reward signal, fluency improves for the model, and annotators prefer the Reward-**EARD** in 67% of the cases for fluency.

### 6.7.5 Impact of character and positional embeddings

We perform an ablation study to identify the impact of various word embeddings used in **EARD**. When character embedding is not used in **EARD**, the performance on SQuAD sentence-level drops from 18.16 to 17.97 (BLEU-4 scores). Similarly, when positional embeddings are dropped, the performance decreases from 18.16 to 17.87 (BLEU-4 scores).

---

**Passage:** McLetchie was elected on the Lothian regional list and the Conservatives suffered a net loss of five seats , with leader **Annabel Goldie** **claiming that their support had held firm**, nevertheless, she too announced she would step down as leader of the party.

---

**Questions**

*True:* Who announced she would step down as leader of the Conservatives ?

*EARD:* who **claiming that their support had held firm** ?

*Reward-EARD:* who was the leader of the conservatives?

---

Table 6.20: An example where Reward-EARD(Originality) is better than EARD.

### 6.7.6 Case Study: Originality of the Questions

We observe that current state-of-the-art models perform very well in BLEU/QBLEU scores when the actual question significantly overlaps with the passage. For example, consider a passage from the SQuAD dataset in Table 6.19, where except the question word *who*, the model sequentially copies everything from the passage and achieves a QBLEU score of 92.4. However, the model performs poorly in situations where the true question is novel and does not contain many words from the passage itself. To quantify this, we first sort the reference questions based on their BLEU-2 overlap with the passage in ascending order. We then select the first  $N$  reference questions and compute the QBLEU score with the generated questions. The results are shown in orange in Figure 6.5. Towards the left, where there are reference questions with low overlap with the passage, the performance is poor, but it gradually improves as the overlap increases.

The task of generating questions with high *originality* (where the model phrases the question in its own words) is a challenging aspect of AQG since it requires a complete understating of the semantics and syntax of the language. As a first step to improve questions generated on *originality*, we explicitly reward our model for having a low  $n$ -gram score with the passage compared to the initial draft. As a result, we observe that with Reward-**EARD**(Originality), there is an improvement in the performance where the overlap with the passage was less (as shown in blue in Figure 6.5). As shown in Table 6.20, although both questions are answerable given the passage, the question generated from Reward-**EARD**(Originality) is better.

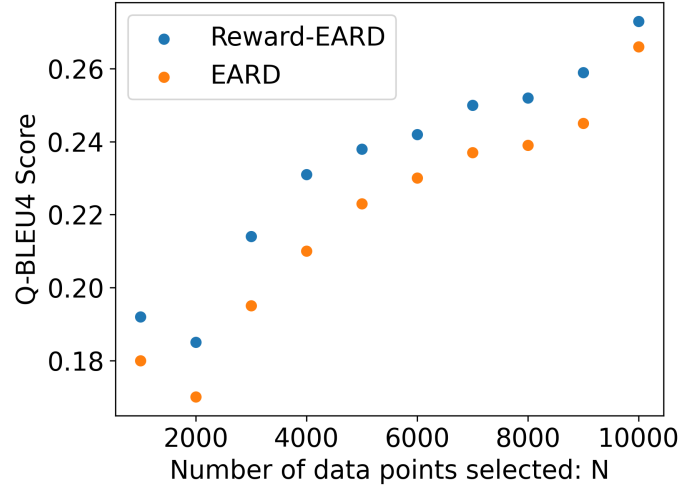


Figure 6.5: Originality Analysis: Plot of Q-BLEU score vs  $N$  - the number of samples selected.

## 6.8 Summary

The main aim of this chapter was to enhance the answerability of the questions generated by seq2seq based AQG systems. To this end, we took the first step of objectively examining the utility of existing metrics for AQG regarding answerability. Specifically, we wanted to see if existing metrics account for the *answerability* of the generated questions. To do so, we took noisy generated questions from three different tasks, *viz.*, document QA, knowledge base QA and visual QA, and showed that the *answerability* scores assigned by humans did not correlate well with existing metrics. Based on these studies, we proposed modifying the existing metrics and showed that the proposed Q-Metrics correlate better with human judgments. The proposed Q-Metric involves learnable weights which can be tuned (depending on the source) using human judgments. Finally, we propose an extrinsic evaluation to assess the end utility of these metrics in selecting good AQG systems for creating training data for QA systems.

We then focused on enhancing AQG systems with our proposed encode-attend-refine-decode paradigm to improve specific aspects of the questions: *answerability*, *originality*, and *fluency*. Our proposed model outperforms the existing state-of-the-art models on the SQuAD, HOTPOT-QA, and DROP datasets. Along with automated evaluations, we also conducted human evaluations to validate our findings. We further showed that using Reward-**EARD** improves the initial draft on desired properties, such

as fluency, answerability, and originality.



## CHAPTER 7

### Conclusion and Future Work

In this thesis, we focused on certain limitations in NLG systems that surfaced in the RNN-based seq2seq models during the third phase of the evolution of NLG. Specifically, we focused on i) repeating phrases, which were a common occurrence across NLG tasks, ii) rendering inadequate descriptions from structured data, and iii) generating questions with limited answerability from a given context.

We hypothesized that across these limitations, a common solution of enriching the contextual representations can be beneficial. To this end, we proposed the *encode-attend-refine-decode* paradigm for RNN-based seq2seq models, where the *refine* module enriches contextual representations. For each of the limitations mentioned above, we propose a seq2seq model based on *encode-attend-refine-decode* paradigm and observe improvements as compared to the respective baseline models based on the *encode-attend-decode* paradigm.

Our first model focused on avoiding repeating phrases for the task of query-based abstractive summarization. We enrich the contextual representation in the refine module by diversifying the context vector computed by the attend module. The diversification is achieved by orthogonalizing the context vector to either its immediate predecessor or the history of context vectors. Moreover, we add a learnable gate that computes the fraction of previous context vectors to be removed from the corresponding context vector. Additionally, as there was no large-scale dataset for query-based abstractive summarization available prior to our work, we introduced a new dataset for this task. The new dataset is based on Debatepedia, and consists of 12,695  $\{passage, query, summary\}$  triplets. We empirically verify that our proposed model leads to higher gains as compared to several baselines. Our model gives an absolute improvement of 12.62 points on the ROUGE-L score compared to its corresponding encode-attend-decode based model. Our model also outperforms the baseline method based on the encode-attend-refine-decode paradigm by 8.66 points on the ROUGE-L score.

Our second model focused on improving the adequacy of the rendered descriptions from structured data. We observed that certain task-specific aspects such as *stay-on* and *never-look-back* properties exhibited by a human while writing a description were not explicitly modeled in vanilla seq2seq models. Therefore, we model these task-specific aspects in our refine module by introducing a learnable gate that learnt to toggle between *stay-on* and *never-look-back* properties while generating the next word at time step  $t$ . Second, the *never-look-back* is modeled using a gated orthogonalization mechanism. We thus enrich the context vector generated from the attend module by modeling these two aspects. We validated our model on a large scale WIKIBIO dataset for English (Lebret *et al.*, 2016), French and German languages (Shetty M, 2018). We empirically verified that our model outperforms a model based on the encode-attend-decode paradigm giving 1.96%, 12.46%, 14.47% relative improvement on BLEU-4 for English, French and German, respectively. Further, through human evaluations, we observed that encoding such task-specific characteristics indeed improved the adequacy and fluency of the models.

Our third model focused on improving the answerability of automatically generated questions. We observed that before improving an AQG system with respect to answerability, it is crucial to quantify the *answerability* of a question. Thus, as a first step, we conducted a thorough human study and examined the current  $n$ -gram based evaluation metrics used for evaluating AQG systems. Based on these studies, we inferred that these metrics do not correlate with the answerability scores assigned by human annotators. Therefore, we proposed a new Q-metric with learnable weights, which are tuned using the human judgments collected as a part of our study. We showed that the proposed Q-metric correlates better with human judgments as compared to the  $n$ -gram metrics. Based on the above proposed metric, we focused on improving AQG systems to generate more answerable questions. More specifically, we generate the question in two passes. Our Preliminary Decoder generates an initial draft which our Refinement Decoder then refines. Our attention module consists of three networks: **Encoder-to-PreliminaryDecoder**, **PreliminaryDecoder-to-RefinementDecoder**, and **Encoder-to-RefinementDecoder** to assist the generation in two passes. The former one assists the Preliminary Decoder in generating an initial draft. The remaining two assist the Refinement Decoder in generating the *refined* version while attending to the

initial draft and the passage simultaneously. Also, to explicitly improve the answerability of the question in the revised draft, we further fine-tune our model using REINFORCE algorithm with a baseline (Williams, 1992) and provide explicit feedback on answerability using our proposed Q-metric. Our proposed model outperforms the corresponding baselines based on encode-attend-decode by 10.9, 9.54, 5.66% (Q-BLEU-4) and 7.83, 7.57, 15.25 % (BLEU-4) on SQuAD, HOTPOT-QA, and DROP datasets, even when no explicit reward is given for answerability. With further fine-tuning, our model’s performance increases by approximately 2% on Q-BLEU4 across these datasets.

## 7.1 Future Directions

This section presents some of the future directions that could be pursued based on our work.

*Tackling other limitations:* Apart from the limitations highlighted in this thesis, there are other limitations, such as hallucinations (Tian *et al.*, 2019; Nie *et al.*, 2019; Maynez *et al.*, 2020; Rohrbach *et al.*, 2018) in generated text, incomplete summaries in abstractive summarization (Gehrmann *et al.*, 2018; Chen and Bansal, 2018; Saito *et al.*, 2020), shallow questions being generated in question generation (Gao *et al.*, 2018; Ma *et al.*, 2020b; Pan *et al.*, 2020), etc. It is obvious that basic seq2seq models are not explicitly designed to address such limitations. As a future direction, various strategies could be adopted in the refine module to enrich the contextual representations to overcome these limitations.

*Fusing solutions for task-agnostic and task-specific problems:* In this thesis, we proposed different models, to avoid the task-agnostic problem of repeating phrases in Chapter 4, and task-specific characteristics in Chapters 5 and 6. However, repeating phrases could still occur while rendering descriptions from structured data and while generating questions. It is not straightforward to combine the refine modules proposed for individually addressing these limitations as our solution of diversifying the context vector will affect the *stay-on* property for rendering descriptions from structured data. One solution could be to add another module in the *encode-attend-refine-decode* paradigm that further optimizes the representations to avoid task agnostic problems such

as repeating phrases, hallucinations, *etc.*

*Adopting encode-attend-refine-decode paradigm for Transformer based seq2seq models:* Although the limitations highlighted in this thesis were discussed in the context of RNN-based seq2seq models, some of these problems, such as repeating phrases (Li *et al.*, 2019; Welleck *et al.*, 2019; Jiang *et al.*, 2020) and inadequate descriptions (Parikh *et al.*, 2020; Puduppully and Lapata, 2021; Rebuffel *et al.*, 2020; Xiao and Wang, 2021) from structured data are still observed in transformer based seq2seq models also. As a future direction, the proposed strategies could be adopted and validated for transformer-based seq2seq models also.

*Proposing task-specific metrics for other NLG tasks:* In this work, we proposed a metric for the task of AQG. Similar metrics could also be proposed for other NLG tasks instead of adopting metrics that were developed for MT. A recent survey Sai *et al.* (2020) also highlights the need for such task-specific metrics. In particular, they point out that it is not prudent to blindly adopt metrics that were developed for MT for other NLG tasks as such metrics may not focus on certain aspects of the generated output which are specific to that task (*e.g.*, BLEU does not capture coverage which is important for the task of generating descriptions from structured data).

## REFERENCES

1. **Agarap, A. F.** (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.
2. **Ali, H., Y. Chali, and S. A. Hasan**, Automation of question generation from sentences. *In Proceedings of QG2010: The Third Workshop on Question Generation*. Citeseer, 2010.
3. **Angeli, G., P. Liang, and D. Klein**, A simple domain-independent probabilistic approach to generation. *In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10*. Association for Computational Linguistics, Stroudsburg, PA, USA, 2010. URL <http://dl.acm.org/citation.cfm?id=1870658.1870707>.
4. **Antol, S., A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh**, VQA: Visual Question Answering. *In International Conference on Computer Vision (ICCV)*. 2015.
5. **Bahdanau, D., K. Cho, and Y. Bengio** (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, **abs/1409.0473**.
6. **Bao, J., D. Tang, N. Duan, Z. Yan, Y. Lv, M. Zhou, and T. Zhao** (2018). Table-to-text: Describing table region with natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, **32**(1). URL <https://ojs.aaai.org/index.php/AAAI/article/view/11944>.
7. **Barzilay, R. and M. Lapata**, Collective content selection for concept-to-text generation. *In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05*. Association for Computational Linguistics, Stroudsburg, PA, USA, 2005. URL <http://dx.doi.org/10.3115/1220575.1220617>.
8. **Bastings, J. and K. Filippova** (2020). The elephant in the interpretability room: Why use attention as explanation when we have saliency methods?
9. **Basu, S., G. S. Ramachandran, N. S. Keskar, and L. R. Varshney** (2021). Mirostat: A neural text decoding algorithm that directly controls perplexity.
10. **Baumel, T., M. Eyal, and M. Elhadad** (2018). Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *CoRR*, **abs/1801.07704**.
11. **Belz, A.** (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.*, **14**(4), 431–455. ISSN 1351-3249. URL <http://dx.doi.org/10.1017/S1351324907004664>.

12. **Belz, A.** and **E. Kow**, System building cost vs. output quality in data-to-text generation. In *Proceedings of the 12th European Workshop on Natural Language Generation, ENLG '09*. Association for Computational Linguistics, Stroudsburg, PA, USA, 2009. URL <http://dl.acm.org/citation.cfm?id=1610195.1610198>.
13. **Ben-younes, H., R. Cadene, M. Cord,** and **N. Thome**, Mutan: Multimodal tucker fusion for visual question answering. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 1. 2017.
14. **Bengio, Y., P. Simard,** and **P. Frasconi** (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**(2), 157–166.
15. **Blatz, J., E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. San-  
chis,** and **N. Ueffing**, Confidence estimation for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, 2004.
16. **Bosma, W.** (2005). Query-based summarization using rhetorical structure theory. *LOT Occasional Series*, **4**, 29–44.
17. **Boyd, S.**, Trend: A system for generating intelligent descriptions of time-series data. In *In Proceedings of the IEEE International Conference on Intelligent Processing Systems (ICIPS-1998)*. 1998.
18. **Brown, P. F., J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Laf-  
ferty, R. L. Mercer,** and **P. S. Roossin** (1990). A statistical approach to machine trans-  
lation. *Computational Linguistics*, **16**(2), 79–85. URL <https://www.aclweb.org/anthology/J90-2002>.
19. **Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Nee-  
lakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger,  
T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse,  
M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. Mc-  
Candlish, A. Radford, I. Sutskever,** and **D. Amodei** (2020). Language models are  
few-shot learners.
20. **Callison-Burch, C.**, Fast, cheap, and creative: Evaluating translation quality using  
amazon’s mechanical turk. In *Proceedings of the 2009 Conference on Empirical Meth-  
ods in Natural Language Processing: Volume 1 - Volume 1, EMNLP '09*. Association  
for Computational Linguistics, Stroudsburg, PA, USA, 2009. ISBN 978-1-932432-59-  
6.
21. **Callison-Burch, C., M. Osborne,** and **P. Koehn**, Re-evaluation the role of bleu in  
machine translation research. In *EACL*. The Association for Computer Linguistics,  
2006.
22. **Cao, Z., W. Li, S. Li, F. Wei,** and **Y. Li**, Attsum: Joint learning of focusing and sum-  
marization with neural attention. In *Proceedings of COLING 2016, the 26th Inter-  
national Conference on Computational Linguistics: Technical Papers*. The COLING  
2016 Organizing Committee, Osaka, Japan, 2016. URL [http://aclweb.org/  
anthology/C16-1053](http://aclweb.org/anthology/C16-1053).

23. **Chan, Y.-H.** and **Y.-C. Fan**, A recurrent bert-based model for question generation. *In Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. 2019.
24. **Chen, D. L.** and **R. J. Mooney**, Learning to sportscast: A test of grounded language acquisition. *In Proceedings of the 25th International Conference on Machine Learning, ICML '08*. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-205-4. URL <http://doi.acm.org/10.1145/1390156.1390173>.
25. **Chen, Q., X. Zhu, Z. Ling, S. Wei,** and **H. Jiang**, Distraction-based neural networks for modeling documents. *In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. 2016.
26. **Chen, Y.-C.** and **M. Bansal** (2018). Fast abstractive summarization with reinforce-selected sentence rewriting.
27. **Chen, Z., W. Chen, H. Zha, X. Zhou, Y. Zhang, S. Sundaresan,** and **W. Y. Wang** (2020). Logic2text: High-fidelity natural language generation from logical forms.
28. **Cho, K., B. van Merriënboer, D. Bahdanau,** and **Y. Bengio** (2014). On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, **abs/1409.1259**. URL <http://arxiv.org/abs/1409.1259>.
29. **Chopra, S., M. Auli,** and **A. M. Rush**, Abstractive sentence summarization with attentive recurrent neural networks. *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, 2016a. URL <http://www.aclweb.org/anthology/N16-1012>.
30. **Chopra, S., M. Auli, A. M. Rush,** and **S. Harvard** (2016b). Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16*, 93–98.
31. **Chorowski, J.** and **N. Jaitly**, Towards better decoding and language model integration in sequence to sequence models. 2017. URL <https://arxiv.org/abs/1612.02695>.
32. **Cohen, J.** (1968). Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological bulletin*, **70**(4), 213—220. ISSN 0033-2909.
33. **Conroy, J. M.** and **D. P. O’leary**, Text summarization via hidden markov models. SIGIR ’01. Association for Computing Machinery, New York, NY, USA, 2001. ISBN 1581133316. URL <https://doi.org/10.1145/383952.384042>.
34. **Dale, R., S. Geldof,** and **J.-P. Prost**, Coral : Using natural language generation for navigational assistance. *In M. J. Oudshoorn* (ed.), *Twenty-Sixth Australasian Computer Science Conference (ACSC2003)*, volume 16 of *CRPIT*. ACS, Adelaide, Australia, 2003.
35. **Daumé III, H.** (2009). Bayesian query-focused summarization. *CoRR*, **abs/0907.1814**.

36. **Deng, Y., W. Zhang, Y. Li, M. Yang, W. Lam, and Y. Shen**, Bridging hierarchical and sequential context modeling for question-driven extractive answer summarization. SIGIR '20. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450380164. URL <https://doi.org/10.1145/3397271.3401208>.
37. **Denkowski, M. and A. Lavie**, Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*. 2014.
38. **Doddington, G.**, Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research, HLT '02*. 2002.
39. **Dong, L., N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H. Hon** (2019). Unified language model pre-training for natural language understanding and generation. *CoRR*, [abs/1905.03197](https://arxiv.org/abs/1905.03197).
40. **Du, X. and C. Cardie**, Identifying where to focus in reading comprehension for neural question generation. In *EMNLP*. Association for Computational Linguistics, 2017.
41. **Du, X., J. Shao, and C. Cardie**, Learning to ask: Neural question generation for reading comprehension. In *ACL (1)*. Association for Computational Linguistics, 2017.
42. **Dua, D., Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner**, DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*. 2019.
43. **Duan, N., D. Tang, P. Chen, and M. Zhou**, Question generation for question answering. In *EMNLP*. Association for Computational Linguistics, 2017.
44. **Duboue, P. A. and K. R. McKeown**, Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. 2003. URL <https://www.aclweb.org/anthology/W03-1016>.
45. **Duchi, J., E. Hazan, and Y. Singer** (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, **12**(7).
46. **Edmundson, H. P.** (1969). New methods in automatic extracting. *Journal of the ACM (JACM)*, **16**(2), 264–285.
47. **Elhadad, M. and J. Robin**, An overview of SURGE: a reusable comprehensive syntactic realization component. In *Eighth International Natural Language Generation Workshop (Posters and Demonstrations)*. 1996. URL <https://www.aclweb.org/anthology/W96-0501>.
48. **Feigenblat, G., H. Roitman, O. Boni, and D. Konopnicki**, Unsupervised query-focused multi-document summarization using the cross entropy method. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17*. Association for Computing Machinery, New York, NY, USA, 2017. ISBN 9781450350228. URL <https://doi.org/10.1145/3077136.3080690>.



49. **Fuentes, M., E. Alfonseca, and H. Rodríguez**, Support vector machines for query-focused summarization trained and evaluated on pyramid data. *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Association for Computational Linguistics, Prague, Czech Republic, 2007. URL <https://www.aclweb.org/anthology/P07-2015>.
50. **Galanis, D. and I. Androutsopoulos**, Generating multilingual descriptions from linguistically annotated owl ontologies: The naturalowl system. *In Proceedings of the Eleventh European Workshop on Natural Language Generation, ENLG '07*. Association for Computational Linguistics, Stroudsburg, PA, USA, 2007. URL <http://dl.acm.org/citation.cfm?id=1610163.1610188>.
51. **Gao, Y., P. Li, I. King, and M. R. Lyu**, Interconnected question generation with coreference alignment and conversation flow modeling. *In ACL (1)*. Association for Computational Linguistics, 2019.
52. **Gao, Y., J. Wang, L. Bing, I. King, and M. R. Lyu** (2018). Difficulty controllable question generation for reading comprehension. *CoRR*, **abs/1807.03586**.
53. **Gates, D.**, Generating look-back strategy questions from expository texts. *In The Workshop on the Question Generation Shared Task and Evaluation Challenge, NSF, Arlington, VA*. <http://www.cs.memphis.edu/~vrus/questiongeneration//1-Gates-QG08.pdf>. 2008.
54. **Gatt, A. and A. Belz** (2010). Introducing shared tasks to nlg: The tuna shared task evaluation challenges, 264–293.
55. **Gehrmann, S., Y. Deng, and A. M. Rush** (2018). Bottom-up abstractive summarization. *CoRR*, **abs/1808.10792**.
56. **Geng, X., X. Feng, B. Qin, and T. Liu**, Adaptive multi-pass decoder for neural machine translation. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 2018. URL <https://www.aclweb.org/anthology/D18-1048>.
57. **Gers, F. A., J. Schmidhuber, and F. Cummins**, Learning to forget: continual prediction with lstm. *In 1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2. 1999.
58. **Goldberg, E., N. Driedger, and R. I. Kittredge** (1994). Using natural-language processing to produce weather forecasts. *IEEE Expert*, **9(2)**, 45–53.
59. **Green, N.**, Generation of biomedical arguments for lay readers. *In Proceedings of the Fourth International Natural Language Generation Conference, INLG '06*. Association for Computational Linguistics, Stroudsburg, PA, USA, 2006. ISBN 1-932432-72-8. URL <http://dl.acm.org/citation.cfm?id=1706269.1706292>.
60. **Gu, J., Z. Lu, H. Li, and V. O. Li**, Incorporating copying mechanism in sequence-to-sequence learning. *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 2016. URL <http://www.aclweb.org/anthology/P16-1154>.

61. **Gu, J., M. Mirshekari, Z. Yu, and A. Sisto** (2021). Chaincqq: Flow-aware conversational question generation.
62. **Harrison, V. and M. Walker** (2018). Neural generation of diverse questions using answer focus, contextual and linguistic features. *arXiv preprint arXiv:1809.02637*.
63. **Hasselqvist, J., N. Helmertz, and M. Kågebäck** (2017). Query-based abstractive summarization using neural networks. *CoRR*, **abs/1712.06100**.
64. **Heilman, M. and N. A. Smith**, Good question! statistical ranking for question generation. In *HLT-NAACL*. The Association for Computational Linguistics, 2010.
65. **Hermann, K. M., T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom**, Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 2015.
66. **Hinton, G.** (2016). Root mean square error propagation.
67. **Hochreiter, S. and J. Schmidhuber** (1997). Long short-term memory. *Neural Comput.*, **9**(8), 1735–1780. ISSN 0899-7667.
68. **Holtzman, A., J. Buys, M. Forbes, and Y. Choi** (2019). The curious case of neural text degeneration. *CoRR*, **abs/1904.09751**.
69. **Hu, B., Q. Chen, and F. Zhu** (2015). Lcsts: A large scale chinese short text summarization dataset. *arXiv preprint arXiv:1506.05865*.
70. **Hutchins, W. J., L. Dostert, and P. Garvin**, The georgetown-i.b.m. experiment. In *In*. John Wiley Sons, 1955.
71. **Iso, H., Y. Uehara, T. Ishigaki, H. Noji, E. Aramaki, I. Kobayashi, Y. Miyao, N. Okazaki, and H. Takamura**, Learning to select, track, and generate for data-to-text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 2019. URL <https://www.aclweb.org/anthology/P19-1202>.
72. **Jain, U., Z. Zhang, and A. G. Schwing**, Creativity: Generating diverse questions using variational autoencoders. In *CVPR*. IEEE Computer Society, 2017.
73. **Jia, R. and P. Liang** (2017). Adversarial examples for evaluating reading comprehension systems. *CoRR*, **abs/1707.07328**.
74. **Jiang, S., T. Wolf, C. Monz, and M. de Rijke** (2020). Tldr: Token loss dynamic reweighting for reducing repetitive utterance generation. *arXiv preprint arXiv:2003.11963*.
75. **Kalady, S., A. Elikkottil, and R. Das**, Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, volume 2. questiongeneration.org, 2010.
76. **Kiddon, C., L. Zettlemoyer, and Y. Choi**, Globally coherent text generation with neural checklist models. In *EMNLP*. The Association for Computational Linguistics, 2016.

77. **Kim, J.** and **R. J. Mooney**, Generative alignment and semantic parsing for learning from ambiguous supervision. *In Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10. Association for Computational Linguistics, Stroudsburg, PA, USA, 2010. URL <http://dl.acm.org/citation.cfm?id=1944566.1944628>.
78. **Kim, Y., H. Lee, J. Shin,** and **K. Jung** (2018). Improving neural question generation using answer separation. *CoRR*, **abs/1809.02393**.
79. **Kingma, D.** and **J. Ba** (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
80. **Koehn, P.**, Pharaoh: a beam search decoder for phrase-based statistical machine translation models. *In Conference of the Association for Machine Translation in the Americas*. Springer, 2004.
81. **Koehn, P., F. J. Och,** and **D. Marcu**, Statistical phrase-based translation. *In Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 2003. URL <https://www.aclweb.org/anthology/N03-1017>.
82. **Konstas, I.** and **M. Lapata**, Unsupervised concept-to-text generation with hypergraphs. *In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Montréal, Canada, 2012. URL <https://www.aclweb.org/anthology/N12-1093>.
83. **Konstas, I.** and **M. Lapata** (2013a). A global model for concept-to-text generation. *J. Artif. Int. Res.*, **48**(1), 305–346. ISSN 1076-9757.
84. **Konstas, I.** and **M. Lapata**, Inducing document plans for concept-to-text generation. *In EMNLP. ACL, 2013b*. ISBN 978-1-937284-97-8. URL <http://dblp.uni-trier.de/db/conf/emnlp/emnlp2013.html#KonstasL13>.
85. **Kulkarni, S., S. Chammas, W. Zhu, F. Sha,** and **E. Ie** (2020). Aquamuse: Automatically generating datasets for query-based multi-document summarization.
86. **Kumar, V., Y. Hua, G. Ramakrishnan, G. Qi, L. Gao,** and **Y.-F. Li**, Difficulty-controllable multi-hop question generation from knowledge graphs. *In C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, and **F. Gandon** (eds.), *The Semantic Web – ISWC 2019*. Springer International Publishing, Cham, 2019.*
87. **Kunichika, H., T. Katayama, T. Hirashima,** and **A. Takeuchi**, Automated question generation methods for intelligent english learning systems and its evaluation. *In Proc. of ICCE*. 2004.
88. **Kurdi, G., J. Leo, B. Parsia, U. Sattler,** and **S. Al-Emari** (2020). A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, **30**(1), 121–204.

89. **Labutov, I., S. Basu, and L. Vanderwende**, Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 2015. URL <https://www.aclweb.org/anthology/P15-1086>.
90. **Lai, G., Q. Xie, H. Liu, Y. Yang, and E. H. Hovy**, RACE: large-scale reading comprehension dataset from examinations. In *EMNLP*. Association for Computational Linguistics, 2017.
91. **Laskar, M. T. R., E. Hoque, and J. X. Huang** (2020). Wsl-ds: Weakly supervised learning with distant supervision for query focused multi-document abstractive summarization.
92. **Lavie, A., K. Sagae, and S. Jayaraman**, The significance of recall in automatic metrics for mt evaluation. In *AMTA*. 2004.
93. **Lebret, R., 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*. Association for Computational Linguistics, Austin, Texas, 2016. URL <https://aclweb.org/anthology/D16-1128>.
94. **Lee, D. B., S. Lee, W. T. Jeong, D. Kim, and S. J. Hwang**, Generating diverse and consistent QA pairs from contexts with information-maximizing hierarchical conditional VAEs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2020. URL <https://www.aclweb.org/anthology/2020.acl-main.20>.
95. **Levy, R. and G. Andrew**, Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA), Genoa, Italy, 2006. URL [http://www.lrec-conf.org/proceedings/lrec2006/pdf/513\\_pdf.pdf](http://www.lrec-conf.org/proceedings/lrec2006/pdf/513_pdf.pdf).
96. **Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer** (2019). BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, **abs/1910.13461**.
97. **Li, M., S. Roller, I. Kulikov, S. Welleck, Y. Boureau, K. Cho, and J. Weston** (2019). Don't say that! making inconsistent dialogue unlikely with unlikelihood training. *CoRR*, **abs/1911.03860**.
98. **Li, P., W. Lam, L. Bing, W. Guo, and H. Li**, Cascaded attention based unsupervised information distillation for compressive summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 2017a. URL <https://www.aclweb.org/anthology/D17-1221>.
99. **Li, Y., N. Duan, B. Zhou, X. Chu, W. Ouyang, and X. Wang** (2017b). Visual question generation as dual task of visual question answering. *CoRR*, **abs/1709.07192**.

100. **Li, Y., N. Duan, B. Zhou, X. Chu, W. Ouyang, X. Wang, and M. Zhou**, Visual question generation as dual task of visual question answering. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
101. **Liang, P., M. I. Jordan, and D. Klein**, Learning semantic correspondences with less supervision. *In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09. Association for Computational Linguistics, Stroudsburg, PA, USA, 2009. ISBN 978-1-932432-45-9. URL <http://dl.acm.org/citation.cfm?id=1687878.1687893>.
102. **Lin, C.-Y.**, Rouge: A package for automatic evaluation of summaries. *In Proc. ACL workshop on Text Summarization Branches Out*. 2004.
103. **Lin, C.-Y.**, Automatic question generation from queries. Citeseer, 2008.
104. **Lindberg, D., F. Popowich, J. Nesbit, and P. Winne**, Generating natural language questions to support learning on-line. *In Proceedings of the 14th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Sofia, Bulgaria, 2013. URL <https://www.aclweb.org/anthology/W13-2114>.
105. **Liu, B., H. Wei, D. Niu, H. Chen, and Y. He**, Asking questions the human way: Scalable question-answer generation from text corpus. *In Proceedings of The Web Conference 2020, WWW '20*. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450370233. URL <https://doi.org/10.1145/3366423.3380270>.
106. **Liu, C., R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau**, How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *In EMNLP*. The Association for Computational Linguistics, 2016.
107. **Liu, M., R. A. Calvo, and V. Rus**, Automatic question generation for literature review writing support. *In V. Aleven, J. Kay, and J. Mostow (eds.), Intelligent Tutoring Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-13388-6.
108. **Liu, T., F. Luo, Q. Xia, S. Ma, B. Chang, and Z. Sui (2019a)**. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. *Proceedings of the AAAI Conference on Artificial Intelligence*, **33**(01), 6786–6793. URL <https://ojs.aaai.org/index.php/AAAI/article/view/4653>.
109. **Liu, T., K. Wang, L. Sha, B. Chang, and Z. Sui (2017a)**. Table-to-text generation by structure-aware seq2seq learning. *CoRR*, **abs/1711.09724**.
110. **Liu, T., K. Wang, L. Sha, B. Chang, and Z. Sui (2017b)**. Table-to-text generation by structure-aware seq2seq learning. *CoRR*, **abs/1711.09724**. URL <http://arxiv.org/abs/1711.09724>.
111. **Liu, Y. (2019)**. Fine-tune BERT for extractive summarization. *CoRR*, **abs/1903.10318**.
112. **Liu, Y. and M. Lapata (2019)**. Text summarization with pretrained encoders. *CoRR*, **abs/1908.08345**.

113. **Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov** (2019b). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, **abs/1907.11692**.
114. **Liu, Y., S.-h. Zhong, and W. Li**, Query-oriented multi-document summarization via unsupervised deep learning. *In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26. 2012.
115. **Loper, E. and S. Bird**, Nltk: The natural language toolkit. *In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, ETMTNLP '02*. 2002.
116. **Lopez, L. E., D. K. Cruz, J. C. B. Cruz, and C. Cheng** (2020). Transformer-based end-to-end question generation. *arXiv preprint arXiv:2005.01107*.
117. **Lopyrev, K.** (2015). Generating news headlines with recurrent neural networks. *arXiv preprint arXiv:1512.01712*.
118. **Luhn, H. P.** (1958). The automatic creation of literature abstracts. *IBM Journal of research and development*, **2**(2), 159–165.
119. **Luong, T., I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba**, Addressing the rare word problem in neural machine translation. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 2015. URL <http://www.aclweb.org/anthology/P15-1002>.
120. **Ma, S., Z.-H. Deng, and Y. Yang**, An unsupervised multi-document summarization framework based on neural document model. *In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, 2016. URL <https://www.aclweb.org/anthology/C16-1143>.
121. **Ma, X., Q. Zhu, Y. Zhou, and X. Li** (2020a). Improving question generation with sentence-level semantic matching and answer position inferring. *Proceedings of the AAAI Conference on Artificial Intelligence*, **34**(05), 8464–8471. URL <https://ojs.aaai.org/index.php/AAAI/article/view/6366>.
122. **Ma, X., Q. Zhu, Y. Zhou, X. Li, and D. Wu** (2020b). Asking complex questions with multi-hop answer-focused reasoning.
123. **Mani, I. and E. Bloedorn** (1997). Multi-document summarization by graph search and matching. *CoRR*, **cmp-lg/9712004**.
124. **Mann, W. C. and S. A. Thompson**, *Rhetorical structure theory: A theory of text organization*. University of Southern California, Information Sciences Institute Los Angeles, 1987.
125. **Mannem, P., R. Prasad, and A. Joshi** (2010). Question generation from paragraphs at upenn: Qgstecc system description.

126. **Maynez, J., S. Narayan, B. Bohnet, and R. McDonald** (2020). On faithfulness and factuality in abstractive summarization.
127. **Mazidi, K. and R. D. Nielsen**, Linguistic considerations in automatic question generation. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, 2014. URL <https://www.aclweb.org/anthology/P14-2053>.
128. **McHugh, M. L.** (2012). Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, **22**(3), 276–282.
129. **Mei, H., M. Bansal, and M. R. Walter**, What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, 2016. URL <http://www.aclweb.org/anthology/N16-1086>.
130. **Mi, H., B. Sankaran, Z. Wang, and A. Ittycheriah**, Coverage embedding models for neural machine translation. *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 2016. URL <https://www.aclweb.org/anthology/D16-1096>.
131. **Miller, A. H., A. Fisch, J. Dodge, A. Karimi, A. Bordes, and J. Weston**, Key-value memory networks for directly reading documents. *In EMNLP*. The Association for Computational Linguistics, 2016.
132. **Mitkov, R. and L. A. Ha**, Computer-aided generation of multiple-choice tests. *In Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*. 2003. URL <https://www.aclweb.org/anthology/W03-0203>.
133. **Mohankumar, A. K., P. Nema, S. Narasimhan, M. M. Khapra, B. V. Srinivasan, and B. Ravindran**, Towards transparent and explainable attention models. *In ACL*. Association for Computational Linguistics, 2020.
134. **Moryossef, A., Y. Goldberg, and I. Dagan**, Step-by-step: Separating planning from realization in neural data-to-text generation. *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 2019. URL <https://www.aclweb.org/anthology/N19-1236>.
135. **Mostow, J. and W. Chen**, Generating instruction automatically for the reading strategy of self-questioning. *In AIED*. 2009.
136. **Nakanishi, M., T. Kobayashi, and Y. Hayashi**, Towards answer-unaware conversational question generation. *In Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Association for Computational Linguistics, Hong Kong, China, 2019. URL <https://www.aclweb.org/anthology/D19-5809>.

137. **Nallapati, R., B. Zhou, C. N. dos Santos, Ç. Gülçehre, and B. Xiang**, Abstractive text summarization using sequence-to-sequence rnns and beyond. *In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016.* 2016. URL <http://aclweb.org/anthology/K/K16/K16-1028.pdf>.
138. **Nastase, V.**, Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. *In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, Honolulu, Hawaii, 2008. URL <https://www.aclweb.org/anthology/D08-1080>.
139. **Nema, P., M. Khapra, A. Laha, and B. Ravindran**, Diversity driven attention model for query-based abstractive summarization. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Vancouver, Canada, 2017.
140. **Nema, P., A. K. Mohankumar, M. M. Khapra, B. V. Srinivasan, and B. Ravindran**, Let's ask again: Refine network for automatic question generation. *In EMNLP/IJCNLP (1).* Association for Computational Linguistics, 2019.
141. **Nema, P., S. Shetty, P. Jain, A. Laha, K. Sankaranarayanan, and M. M. Khapra**, Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* Association for Computational Linguistics, New Orleans, Louisiana, 2018.
142. **Nestrov, Y. E.** (1983). A method for solving the convex programming problem with convergence rate  $o(1/k^2)$ . *Dokl. Akad. Nauk SSSR*, **269**, 543–547. URL <https://ci.nii.ac.jp/naid/10029946121/en/>.
143. **Nguyen, T., M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng**, MS MARCO: A human generated machine reading comprehension dataset. *In Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016..* 2016.
144. **Nie, F., J.-G. Yao, J. Wang, R. Pan, and C.-Y. Lin**, A simple recipe towards reducing hallucination in neural surface realisation. *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Florence, Italy, 2019. URL <https://www.aclweb.org/anthology/P19-1256>.
145. **Offerijns, J., S. Verberne, and T. Verhoef** (2020). Better distractions: Transformer-based distractor generation and multiple choice question filtering. *arXiv preprint arXiv:2010.09598*.
146. **Otterbacher, J., G. Erkan, and D. Radev**, Using random walks for question-focused sentence retrieval. *In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing.* Association for



- Computational Linguistics, Vancouver, British Columbia, Canada, 2005. URL <https://www.aclweb.org/anthology/H05-1115>.
147. **Pan, L., Y. Xie, Y. Feng, T.-S. Chua, and M.-Y. Kan**, Semantic graphs for generating deep questions. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2020. URL <https://www.aclweb.org/anthology/2020.acl-main.135>.
  148. **Papineni, K., S. Roukos, T. Ward, and W. Zhu**, Bleu: a method for automatic evaluation of machine translation. *In ACL*. ACL, 2002.
  149. **Parikh, A., X. Wang, S. Gehrmann, M. Faruqui, B. Dhingra, D. Yang, and D. Das**, ToTTo: A controlled table-to-text generation dataset. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 2020. URL <https://www.aclweb.org/anthology/2020.emnlp-main.89>.
  150. **Parikh, S., A. Sai, P. Nema, and M. Khapra**, Eliminet: A model for eliminating options for reading comprehension with multiple choice questions. *In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 2018. URL <https://doi.org/10.24963/ijcai.2018/594>.
  151. **Pascanu, R., T. Mikolov, and Y. Bengio** (2012). Understanding the exploding gradient problem. *CoRR*, **abs/1211.5063**.
  152. **Pasunuru, R. and M. Bansal**, Multi-reward reinforced summarization with saliency and entailment. *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2018. URL <https://www.aclweb.org/anthology/N18-2102>.
  153. **Paulus, R., C. Xiong, and R. Socher** (2018). A deep reinforced model for abstractive summarization. *CoRR*, **abs/1705.04304**.
  154. **Pennington, J., R. Socher, and C. D. Manning**, Glove: Global vectors for word representation. *In Empirical Methods in Natural Language Processing (EMNLP)*. 2014.
  155. **Puduppully, R., L. Dong, and M. Lapata** (2019). Data-to-text generation with content selection and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, **33**(01), 6908–6915. URL <https://ojs.aaai.org/index.php/AAAI/article/view/4668>.
  156. **Puduppully, R. and M. Lapata** (2021). Data-to-text generation with macro planning.
  157. **Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever** (2018). Improving language understanding by generative pre-training.
  158. **Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever** (2019). Language models are unsupervised multitask learners. *OpenAI blog*, **1**(8), 9.

159. **Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang**, Squad: 100, 000+ questions for machine comprehension of text. *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 2016.
160. **Ranzato, M., S. Chopra, M. Auli, and W. Zaremba**, Sequence level training with recurrent neural networks. *In ICLR (Poster)*. 2016.
161. **Rebuffel, C., L. Soulier, G. Scoutheeten, and P. Gallinari**, A hierarchical model for data-to-text generation. *In ECIR (1)*, volume 12035 of *Lecture Notes in Computer Science*. Springer, 2020.
162. **Reddy, S., D. Raghu, M. M. Khapra, and S. Josh**, Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. *In EACL (1)*. Association for Computational Linguistics, 2017.
163. **Reiter, E., S. Sripada, J. Hunter, J. Yu, and I. Davy** (2005). Choosing words in computer-generated weather forecasts. *Artif. Intell.*, **167**(1-2), 137–169. URL <http://dblp.uni-trier.de/db/journals/ai/ai167.html#ReiterSHYD05>.
164. **Rennie, S. J., E. Marcheret, Y. Mroueh, J. Ross, and V. Goel** (2017). Self-critical sequence training for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1179–1195.
165. **Rohrbach, A., L. A. Hendricks, K. Burns, T. Darrell, and K. Saenko**, Object hallucination in image captioning. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 2018. URL <https://www.aclweb.org/anthology/D18-1437>.
166. **Roitman, H., G. Feigenblat, D. Cohen, O. Boni, and D. Konopnicki**, Unsupervised dual-cascade learning with pseudo-feedback distillation for query-focused extractive summarization. *In Proceedings of The Web Conference 2020*. 2020.
167. **Rokhlenko, O. and I. Szpektor**, Generating synthetic comparable questions for news articles. *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 2013. URL <https://www.aclweb.org/anthology/P13-1073>.
168. **Rothe, A., B. M. Lake, and T. M. Gureckis**, Question asking as program generation. *In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*. Curran Associates Inc., Red Hook, NY, USA, 2017. ISBN 9781510860964.
169. **Rumelhart, D. E., G. E. Hinton, and R. J. Williams** (1986). Learning representations by back-propagating errors. *nature*, **323**(6088), 533–536.
170. **Rush, A. M., S. Chopra, and J. Weston**, A neural attention model for abstractive sentence summarization. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 2015. URL <http://aclweb.org/anthology/D15-1044>.

171. **Sachan, M.** and **E. Xing**, Self-training for jointly learning to ask and answer questions. *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2018. URL <https://www.aclweb.org/anthology/N18-1058>.
172. **Sai, A. B., A. K. Mohankumar**, and **M. M. Khapra** (2020). A survey of evaluation metrics used for nlg systems.
173. **Saito, I., K. Nishida, K. Nishida**, and **J. Tomita** (2020). Abstractive summarization with combination of pre-trained sequence-to-sequence and saliency models.
174. **Sankaran, B., H. Mi, Y. Al-Onaizan**, and **A. Ittycheriah** (2016). Temporal attention model for neural machine translation. *arXiv preprint arXiv:1608.02927*.
175. **Schilder, F.** and **R. Kondadadi**, FastSum: Fast and accurate query-based multi-document summarization. *In Proceedings of ACL-08: HLT, Short Papers*. Association for Computational Linguistics, Columbus, Ohio, 2008. URL <https://www.aclweb.org/anthology/P08-2052>.
176. **Scialom, T., B. Piwowarski**, and **J. Staiano**, Self-attention architectures for answer-agnostic neural question generation. *In Proceedings of the 57th annual meeting of the Association for Computational Linguistics*. 2019.
177. **See, A., P. J. Liu**, and **C. D. Manning**, Get to the point: Summarization with pointer-generator networks. *In ACL*. 2017.
178. **Seo, M. J., A. Kembhavi, A. Farhadi**, and **H. Hajishirzi** (2016). Bidirectional attention flow for machine comprehension. *CoRR*, **abs/1611.01603**.
179. **Serban, I. V., A. García-Durán, Ç. Gülçehre, S. Ahn, S. Chandar, A. C. Courville**, and **Y. Bengio**, Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *In ACL (1)*. The Association for Computer Linguistics, 2016a.
180. **Serban, I. V., A. Sordoni, Y. Bengio, A. Courville**, and **J. Pineau**, Building end-to-end dialogue systems using generative hierarchical neural network models. *In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16*. AAAI Press, 2016b. URL <http://dl.acm.org/citation.cfm?id=3016387.3016435>.
181. **Sha, L., L. Mou, T. Liu, P. Poupart, S. Li, B. Chang**, and **Z. Sui** (2017). Order-planning neural text generation from structured data.
182. **Shetty M, S.** (2018). *Natural Language Generation from Structured Data*. Master's thesis, Indian Institute of Technology Madras.
183. **Singh, M., A. Mishra, Y. Oualil, K. Berberich**, and **D. Klakow**, Long-span language models for query-focused unsupervised extractive text summarization. *In European Conference on Information Retrieval*. Springer, 2018.
184. **Song, L., Z. Wang**, and **W. Hamza** (2017). A unified query-based generative model for question generation and question answering. *CoRR*, **abs/1709.01058**.

185. **Sripada, S. G., E. Reiter, and I. Davy** (2003). Sumtime-mousam: Configurable marine weather forecast generator.
186. **Su, D., T. Yu, and P. Fung** (2021). Improve query focused abstractive summarization by incorporating answer relevance. *CoRR*, **abs/2105.12969**. URL <https://arxiv.org/abs/2105.12969>.
187. **Su, J., S. Wu, D. Xiong, Y. Lu, X. Han, and B. Zhang** (2018). Variational recurrent neural machine translation. *CoRR*, **abs/1801.05119**.
188. **Sultan, M. A., S. Chandel, R. F. Astudillo, and V. Castelli**, On the importance of diversity in question generation for qa. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.
189. **Sun, X., J. Liu, Y. Lyu, W. He, Y. Ma, and S. Wang**, Answer-focused and position-aware neural question generation. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 2018.
190. **Sun, Y., D. Tang, N. Duan, T. Qin, S. Liu, Z. Yan, M. Zhou, Y. Lv, W. Yin, X. Feng, et al.** (2019). Joint learning of question answering and question generation. *IEEE Transactions on Knowledge and Data Engineering*, **32**(5), 971–982.
191. **Suzuki, J. and M. Nagata**, Cutting-off redundant repeating generations for neural abstractive summarization. *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, 2017. URL <https://www.aclweb.org/anthology/E17-2047>.
192. **Tang, D., N. Duan, T. Qin, and M. Zhou** (2017). Question answering and question generation as dual tasks. *CoRR*, **abs/1706.02027**.
193. **Tang, J., L. Yao, and D. Chen**, Multi-topic based query-oriented summarization. *In Proceedings of the 2009 SIAM international conference on data mining*. SIAM, 2009.
194. **Tian, R., S. Narayan, T. Sellam, and A. P. Parikh** (2019). Sticking to the facts: Confident decoding for faithful data-to-text generation. *CoRR*, **abs/1910.08684**.
195. **Tu, Z., Y. Liu, Z. Lu, X. Liu, and H. Li** (2017a). Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, **5**, 87–99. URL <https://www.aclweb.org/anthology/Q17-1007>.
196. **Tu, Z., Y. Liu, L. Shang, X. Liu, and H. Li** (2017b). Neural machine translation with reconstruction. *Proceedings of the AAAI Conference on Artificial Intelligence*, **31**(1). URL <https://ojs.aaai.org/index.php/AAAI/article/view/10950>.
197. **Tu, Z., Z. Lu, Y. Liu, X. Liu, and H. Li**, Modeling coverage for neural machine translation. *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 2016. URL <https://www.aclweb.org/anthology/P16-1008>.

198. **Turner, R., S. Sripada, and E. Reiter**, Generating approximate geographic descriptions. In **E. Krahmer and M. Theune** (eds.), *Empirical Methods in Natural Language Generation*, volume 5790 of *Lecture Notes in Computer Science*. Springer, 2010. ISBN 978-3-642-15572-7. URL <http://dblp.uni-trier.de/db/conf/eacl/enlg2010.html#TurnerSR10>.
199. **Varga, A. and L. A. Ha** (2010). Wlv: a question generation system for the qgstec 2010 task b. *Boyer & Piwek (2010)*, 80–83.
200. **Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin**, Attention is all you need. *In Advances in Neural Information Processing Systems*. 2017.
201. **Vogel, S., H. Ney, and C. Tillmann**, HMM-based word alignment in statistical translation. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*. 1996. URL <https://www.aclweb.org/anthology/C96-2141>.
202. **Wan, X.** (2008). Using only cross-document relationships for both generic and topic-focused multi-document summarizations. *Information Retrieval*, **11**(1), 25–49.
203. **Wan, X. and J. Zhang**, Ctsum: Extracting more certain summaries for news articles. *In Proceedings of the 37th International ACM SIGIR Conference on Research Development in Information Retrieval, SIGIR '14*. Association for Computing Machinery, New York, NY, USA, 2014. ISBN 9781450322577. URL <https://doi.org/10.1145/2600428.2609559>.
204. **Wang, L., H. Raghavan, V. Castelli, R. Florian, and C. Cardie**, A sentence compression based framework to query-focused multi-document summarization. *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 2013. URL <https://www.aclweb.org/anthology/P13-1136>.
205. **Wang, L., Z. Xu, Z. Lin, H. Zheng, and Y. Shen**, Answer-driven deep question generation based on reinforcement learning. *In Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Barcelona, Spain (Online), 2020a. URL <https://www.aclweb.org/anthology/2020.coling-main.452>.
206. **Wang, Q., Z. Zhou, L. Huang, S. Whitehead, B. Zhang, H. Ji, and K. Knight**, Paper abstract writing through editing mechanism. *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2018a. URL <https://www.aclweb.org/anthology/P18-2042>.
207. **Wang, Z., W. Hamza, and R. Florian** (2017). Bilateral multi-perspective matching for natural language sentences. *CoRR*, **abs/1702.03814**.
208. **Wang, Z., A. S. Lan, W. Nie, A. E. Waters, P. J. Grimaldi, and R. G. Baraniuk**, Qgnet: a data-driven question generation model for educational content. *In Proceedings of the Fifth Annual ACM Conference on Learning at Scale*. 2018b.

209. **Wang, Z., X. Wang, B. An, D. Yu, and C. Chen**, Towards faithful neural table-to-text generation with content-matching constraints. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2020b. URL <https://www.aclweb.org/anthology/2020.acl-main.101>.
210. **Wei, F., Y. He, W. Li, and Q. Lu**, A query-sensitive graph-based sentence ranking algorithm for query-oriented multi-document summarization. *In 2008 International Symposiums on Information Processing*. 2008.
211. **Weissenborn, D., G. Wiese, and L. Seiffe**, Making neural QA as simple as possible but not simpler. *In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Association for Computational Linguistics, Vancouver, Canada, 2017. URL <https://www.aclweb.org/anthology/K17-1028>.
212. **Weizenbaum, J.** (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, **9**(1), 36–45.
213. **Welleck, S., I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston** (2019). Neural text generation with unlikelihood training. *CoRR*, **abs/1908.04319**.
214. **Werbos, P. J.** (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, **78**(10), 1550–1560.
215. **Wieting, J., T. Berg-Kirkpatrick, K. Gimpel, and G. Neubig** (2019). Beyond bleu: Training neural machine translation with semantic similarity. *arXiv preprint arXiv:1909.06694*.
216. **Williams, R. J.** (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, **8**(3-4), 229–256. ISSN 0885-6125.
217. **Wiseman, S., S. M. Shieber, and A. M. Rush** (2017). Challenges in data-to-document generation. *CoRR*, **abs/1707.08052**.
218. **Wolfe, J. H.**, Automatic question generation from text - an aid to independent study. *In SIGCSE '76*. 1976.
219. **Xia, Y., F. Tian, L. Wu, J. Lin, T. Qin, N. Yu, and T.-Y. Liu**, Deliberation networks: Sequence generation beyond one-pass decoding. *In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, 1784–1794.
220. **Xiao, D., H. Zhang, Y. Li, Y. Sun, H. Tian, H. Wu, and H. Wang** (2020). ERNIE-GEN: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. *CoRR*, **abs/2001.11314**. URL <https://arxiv.org/abs/2001.11314>.
221. **Xiao, Y. and W. Y. Wang** (2021). On hallucination and predictive uncertainty in conditional language generation.
222. **Xie, Y., T. Zhou, Y. Mao, and W. Chen** (2020). Conditional self-attention for query-based summarization.

223. **Xu, J., Y. Wang, D. Tang, N. Duan, P. Yang, Q. Zeng, M. Zhou, and X. Sun**, Asking clarification questions in knowledge-based question answering. *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2019. URL <https://www.aclweb.org/anthology/D19-1172>.
224. **Ya, J., T. Liu, and L. Guo**, A compare-aggregate model with external knowledge for query-focused summarization. *In Z. Huang, W. Beek, H. Wang, R. Zhou, and Y. Zhang (eds.), Web Information Systems Engineering – WISE 2020*. Springer International Publishing, Cham, 2020. ISBN 978-3-030-62008-0.
225. **Yang, Z., Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le** (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, **abs/1906.08237**.
226. **Yang, Z., P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning**, HotpotQA: A dataset for diverse, explainable multi-hop question answering. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.
227. **Yao, K., G. Zweig, and B. Peng** (2015). Attention with intention for a neural network conversation model. *CoRR*, **abs/1510.08565**. URL <http://arxiv.org/abs/1510.08565>.
228. **Yao, X., G. Bouma, and Y. Zhang** (2012). Semantics-based question generation and implementation. *Dialogue & Discourse*, **3**(2), 11–42.
229. **Yuan, X., T. Wang, Ç. Gülçehre, A. Sordoni, P. Bachman, S. Zhang, S. Subramanian, and A. Trischler**, Machine comprehension by text-to-text neural question generation. *In Rep4NLP@ACL*. Association for Computational Linguistics, 2017.
230. **Zajic, D. M., B. Dorr, J. Lin, and R. Schwartz**, Sentence compression as a component of a multi-document summarization system. *In Proceedings of the 2006 document understanding workshop, New York*. 2006.
231. **Zhang, J., Q. Wu, C. Shen, J. Zhang, J. Lu, and A. van den Hengel** (2017). Asking the difficult questions: Goal-oriented visual question generation via intermediate rewards. *CoRR*, **abs/1711.07614**.
232. **Zhang, S. and M. Bansal** (2019). Addressing semantic drift in question generation for semi-supervised question answering. *arXiv preprint arXiv:1909.06356*.
233. **Zhang, X., J. Su, Y. Qin, Y. Liu, R. Ji, and H. Wang**, Asynchronous bidirectional decoding for neural machine translation. *In AACL*. AAAI Press, 2018.
234. **Zhang, X., J. Zhao, and Y. LeCun**, Character-level convolutional networks for text classification. *In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL <https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf>.

235. **Zhao, S., H. Wang, C. Li, T. Liu, and Y. Guan**, Automatically generating questions from queries for community-based question answering. *In Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, 2011. URL <https://www.aclweb.org/anthology/I11-1104>.
236. **Zhao, Y., X. Ni, Y. Ding, and Q. Ke**, Paragraph-level neural question generation with maxout pointer and gated self-attention networks. *In EMNLP*. 2018.
237. **Zheng, Z., X. Si, E. Y. Chang, and X. Zhu**, K2q: Generating natural language questions from keywords with user refinements. *In Proceedings of the 5th International Joint Conference on Natural Language Processing*. 2011. URL <http://aclweb.org/anthology-new/I/I11/I11-1106.pdf>.
238. **Zhou, Q., N. Yang, F. Wei, C. Tan, H. Bao, and M. Zhou**, Neural question generation from text: A preliminary study. *In NLPCC*. 2017.
239. **Zhou, W., M. Zhang, and Y. Wu**, Question-type driven question generation. *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2019. URL <https://www.aclweb.org/anthology/D19-1622>.
240. **Zhu, H., L. Dong, F. Wei, W. Wang, B. Qin, and T. Liu** (2019). Learning to ask unanswerable questions for machine reading comprehension. *CoRR*, **abs/1906.06045**.



## LIST OF PAPERS BASED ON THESIS

1. Let's Ask Again: Refine Network for Automatic Question Generation - **Preksha Nema, Akash Kumar Mohankumar, Mitesh M. Khapra, Balaji Vasan Srinivasan, Balaraman Ravindran**, *The 2019 Conference on Empirical Methods in Natural Language Processing*, (EMNLP, 2019)
2. Towards a Better Metric for Evaluating Question Generation Systems- **Preksha Nema, Mitesh M. Khapra**, *The 2018 Conference on Empirical Methods in Natural Language Processing*, (EMNLP, 2018)
3. Generating Descriptions from Structured Data Using a Bifocal Attention Mechanism and Gated Orthogonalization - **Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, Mitesh M. Khapra**, *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, (NAACL,2018)
4. Diversity driven Attention Model for Query Based Abstractive Summarization - **Preksha Nema, Mitesh M. Khapra, Anirban Laha, Balaraman Ravindran**, *Association for Computational Linguistics, 2017*, (ACL, 2017 )



## DOCTORAL COMMITTEE

**Chairperson:** Dr. P. Sreenivasa Kumar  
Professor  
Department of Computer Science & Engineering  
Indian Institute of Technology Madras

**Research Advisors:** Dr. Mitesh M. Khapra  
Associate Professor  
Department of Computer Science & Engineering  
Indian Institute of Technology Madras

Dr. Balaraman Ravindran  
Professor  
Department of Computer Science & Engineering  
Indian Institute of Technology Madras

**Members:** Dr. Sutanu Chakraborti  
Associate Professor  
Department of Computer Science & Engineering  
Indian Institute of Technology Madras

Dr. N.S. Narayanaswamy  
Professor  
Department of Computer Science & Engineering  
Indian Institute of Technology Madras

Dr. Kaushik Mitra  
Assistant Professor  
Department of Electrical Engineering  
Indian Institute of Technology Madras



## CURRICULUM VITAE

1. **NAME** : Preksha Nema
2. **DATE OF BIRTH** : February 25<sup>th</sup>, 1990
3. **PERMANENT ADDRESS** : 531-14 Shyam Dujiya Nilay,  
Near Ghadi Chowk, Vijaynagar  
Jabalpur, 482002  
Madhya Pradesh  
Email: preksha.nema9@gmail.com  
Phone: +91-9890283245

### 4. **EDUCATIONAL QUALIFICATIONS**

#### **Bachelor of Technology (B.Tech.)**

- |                    |   |   |
|--------------------|---|---|
| Year of Completion | : | 2012  |
| Institution        | : | Visvesvaraya National Institute of Technology,<br>Nagpur, Maharashtra |
| Specialization     | : | Computer Science and<br>Engineering                                   |