Automatic Fact Extraction from Open-Ended Geometry Questions

Ishadi Jayasinghe

188090K

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science in Computer Science and Engineering

Department of Computer Science & Engineering

University of Moratuwa

Sri Lanka

November 2019

DECLARATION

I, Ishadi Jayasinghe, declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The above candidate has carried out research for the Masters thesis/Dissertation under my supervision.

Name of Supervisor: Dr. Surangika Ranathunga

Signature of the Supervisor: Date:

ABSTRACT

Semantic parsing of geometry problems is the first step towards automated geometry problem solvers. Existing systems for this task heavily depend on language-specific NLP tools and use hard-coded parsing rules. Moreover, these systems produce a static set of facts and record low precision scores. In this study, we present the two-step memory network, a novel neural network architecture for deep semantic parsing of GWPs. Our model is language independent and optimized for low-resource domains. Without using any language-specific NLP tool, our system performs as good as existing systems. We also introduce on-demand fact extraction, where a solver can query the model about entities during the solving stage. This is impossible for existing systems; the set of extracted facts with these systems are static after the parsing stage. This feature alleviates the problem of having an imperfect recall.

We also investigate data augmentation techniques for low resource domains to alleviate the difficulties in applying deep learning techniques in the domain above. We also introduce an enhanced metric for evaluating language generative models alleviating the the limitations of exiting metrics. Analysing the results, we come up with a ranking of models on their suitability to be used o low resource domains

Keywords: Semantic Parsing; Deep Learning; Memory Networks; Generative Adversarial Networks; Temperature Sweep

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor Dr. Surangika Ranathunga for the continuous support given for the success of this research, for her patience, motivation, and immense knowledge. I'm extremely thankful for your advices. I strongly believe that without your tremendous mentorship and advice from the beginning, the present work could have not reached this stage. Your wide knowledge and logical way of thinking have been of great source of inspiration for me.

I wish to thank Ms. Vishaka Nanayakkara and Mr. Nisansa de Silva for their valuable insights and guidance from the very early stage of this research. I would like to thank the entire staff of the Department of Computer Science and Engineering, both academic and non-academic for all their help during the course of this work and for providing me with the resources necessary to conduct my research. This research was supported by the University of Moratuwa Senate Research Grant.

I would like to thank Mr. Moksha Manukantha, Ms. Kulakshi Fernando, Ms. Vijini Liyanage, and my family for all the love and support.

Thank you!

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
GWP	Geometry Word Problem
MCQ	Multiple Choice Question
NLG	Natural Language Generative
DNN	Deep Neural Network
GAN	Generative Adversarial Net
CNN	Convolutional Neural Network
RNN	Recurrent Neural Networks
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
DBN	Deep Belief Net
DAE	Denoising Autoencoder
VAE	Variational Autoencoder
MLE	Maximum Likelihood Estimate
RL	Reinforcement Learning
FC	Focus Controlling
PE	Position Encoding

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Background

Mathematics is a domain that involves high logical thinking and reasoning capabilities. Mathematics involves several disciplines such as Arithmetic, Symbolic, and Geometry. Among these fields, Geometry holds an important place as it help students develop the skills of visualization, critical thinking, intuition, perspective, problem-solving, conjecturing, deductive reasoning, logical argument and proof [4].

A proof problem in geometry (refer figure 1.1) normally involves discovering new facts using a set of given facts and a pool of geometry theorems. Although this type of deductions are common in Mathematics, many students find proof problems difficult. According to a large-scale survey done in USA, it is only about 30% of students completing full-year geometry courses that taught proof reached a 75% mastery level proof writing [4]. The study also found that a significant number from highly performing students also found these problems to be difficult. The main reason behind this scenario can be recognized as the fact that there is no definite pattern to solve these problems. A student must use the theorems with the facts given in the problem with no set pattern or a list of definite steps to follow. Thus, a student needs practice on a fairly large number of problems to gain the required problem solving skills [5]. Also, assessing student answers in geometric theorem proving is a time consuming activity. This high time consumption limits the number of practice problems a teacher can solve and discuss during the class [6].

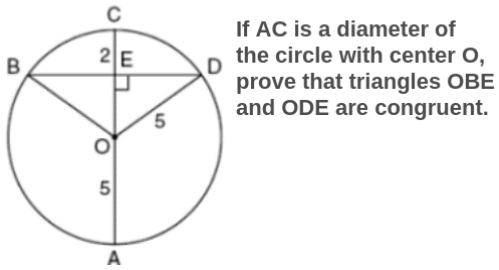


Figure 1.1: An example for a geometry proof problem

1.2 Research Problem

A platform for students to practise geometry problems without the guidance of a teacher can alleviate above issues. There are already implemented systems that partially serve this purpose. MathTutor [6] is a system that can assess a proof written by students. However, this system needs the inputs to be in XML format. Also, it needs the marking scheme of the geometry problem also as an input. Therefore, this system cannot be used by students who are unfamiliar with the input formats required by the system. The usage of the system is also limited by its need for marking schemes.

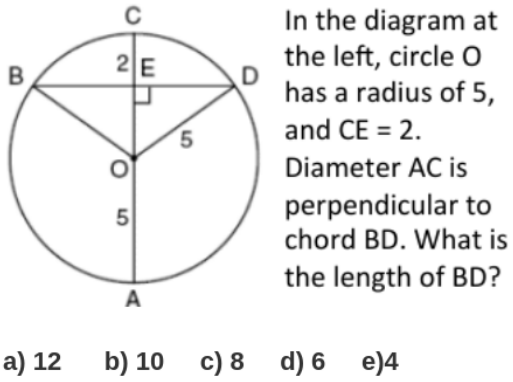


Figure 1.2: An example for an MCQ in geometry

GEOS [3] is a system which does not need such formatted inputs; it can parse the natural language description and any image associated with the geometry problem. However, despite its capability to accept the question in the natural language,GEOS can solve only solve Multiple Choice Questions (MCQs) related

to geometry. An example for a problem of this kind is given in figure 1.2. GEOS solves MCQs by comapring the matching between each choice in the MCQ with the rules it produces parsing the geometry question. Thus, GEOS is neither interpretable nor cannot be used in questions which needs proof. On the other hand, GeoShader [7] is a system which is both interpretable and can be used to generate proofs for geometry problems. GeoShader produces all possible paths for arriving a target result using existing facts and geometry theorems. However, GeoShader also accepts the question in a specific format limiting its usability.

Automatic solving of geometry problems can be modeled as a search problem that traverses through a set of facts [8], or a numerical approach that checks the satisfiability of a set of facts [3]. To use either of these techniques, the set of facts have to be first extracted from the textual description (and the diagram if there is any). As explained above, while most of the research in this line has assumed the existence of a set of facts (a formal machine representation) to reason with [6], little research has looked at parsing a given question in natural language into a formal representation [3, 8].

Shallow forms of semantic parsing usually focus on parsing a sentence to a sequence of word representations such as the case-role assignment of the constituents [9], while deep forms focus on building more formal representations that support automated reasoning. For geometric problem solving, we need the extracted set of facts to support automated reasoning. Thus, deep semantic parsing techniques should be used.

The task of deep parsing of geometry questions is inherently difficult due to several reasons. The inaccurate behavior of existing Natural Language Processing (NLP) tools on this domain can be considered as a critical restriction. For example, Figure 1.3 shows the dependency parse tree produced by the Stanford Dependency parser [1] for the phrase “AB is a chord of the circle with center O.” Even though the word “circle” should be closely related to “O”, the parse tree only shows an indirect path between the two nodes. Another critical problem is the lack of data [3], so using usual deep learning models for this task (due to their data intensiveness) is not viable. Hence, existing systems such as GEOS

Figure 1.3: Dependency parse tree for “AB is a chord of the circle with center O" using Stanford CoreNLP Toolkit [1]

have resorted to using hard-coded parsing rules [3, 8]. This rule-based approach brings forth a couple of limitations. Firstly, this limits the system only for one language, and even within that language, the scalability and the maintenance are difficult as changing or adding new variants of sentences require alterations in those hard-coded rules. These alterations might also require domain-specific knowledge. Secondly, these systems record low precision scores [3].

When solving geometry questions, students do deductive reasoning using axioms and theorems of geometry. [8]. Thus, a system that follows this procedure for arriving at the solution, rather than a numeric approach, is proven to be more user-friendly to the students [8]. Under this approach, a system goes on discovering new relationships until it reaches the target answer. Hence, the importance of entities to the system is dynamic. For an example, during a proof where the goal is to prove the entities AB and CD are equal, and if the system deduces a relationship such as $AB = BC$, proving that BC equals CD would make the system achieve the target. Thus, it will have to explore the facts of BC. However, due to the imperfect recall of the existing systems, the facts relevant to BC can be missing in the extracted fact set. This can inhibit solving the problem.

Despite the limitations introduced by rule-based approaches, one of the main reasons that the existing systems use mostly rule-based approaches is the lack of data in the domain. One option to introduce deep learning techniques to this domain is generating data. Even though an ample amount of text generation models are available [10, 11, 12, 13], all these models are evaluated against large corpora. Moreover, the metrics used to evaluate these models evaluate the generated sample either with respect to the diversity or with respect to the quality. However, a model that produces sentences with a very low quality can produce

those low-quality sentences with a high diversity. Similarly, a model can produce a single high quality sentence over and over again which would result in a high quality, but a very low diversity. So, the results reported for the evaluations of these models are misleading [14]. With these metrics, it is impossible to decide which models perform better in both diversity and quality aspects.

1.3 Research Objectives

Objectives of this research are as follows:

1. Design a language-independent system which can
   (a) extract relations in a higher precision
   (b) extract dynamic number of relations for a given sentence in natural language
   (c) extract facts dynamically about a given entity
   (d) extract relations across multiple sentences
2. Identify and implement language generation mechanism to generate more data from a given small seed corpora so that deep learning techniques can be used on them.

1.4 Contributions

We make the following contributions in this thesis:

- Introducing a novel neural architecture that is optimized for low resource domains, that can be used to parse a geometry problem given in natural language
- Systematic evaluation of existing natural language generation techniques for small seed corpora
- An enhanced metric to evaluate the performance of language generation models on low resource datasets

1.5 Publications

- **Ishadi Jayasinghe**, Surangika Ranathunga "Two-step Memory Networks for Deep Semantic Parsing of Geometry Word Problems" WiML workshop associated with NeurIPS 2019 at Vancouver, BC, Canada ***(Extended Abstract Accepted)***.
  - Have been **awarded** with a travel grant

- **Ishadi Jayasinghe**, Surangika Ranathunga "Two-step Memory Networks for Deep Semantic Parsing of Geometry Word Problems" 46th International Conference on Current Trends in Theory and Practice of Computer Science ***(Accepted)***.
  - Core rank of the conference: B

- **Ishadi Jayasinghe**, Surangika Ranathunga "Evaluating the Performance of Language GANs with Small Seed Corpora" 3rd SLAAI-International Conference on Artificial Intelligence - 2019 ***(Accepted)***.
  - Scopus Indexed

Chapter 2

BACKGROUND

2.1 Overview

A number of deep learning techniques has been used in this study. Some of the most those techniques are discussed in this section. We start with highlighting the importance of Recurrent Neural Networks (RNNs) compared to feed forward networks. Then we discuss the limitations of RNNs in the context of encoder-decoder architectures. There, we show its limitations and how attention mechanisms address them. Even though attention mechanisms alleviate many issues found in RNNs, we show that the models in literature that have employed attention mechanisms have very large capacities, thus making it difficult to be used in low resource domains. We then discuss memory networks, which were initially introduced as a question answering system with the capability to query over multiple sentences considering their temporal order. We show how the issues found in above models are alleviated in memory networks and highlight their ability to provide a good capacity with a comparatively lower amount of trainable parameters. Finally, we conclude this section by discussing Generative Adversarial Nets (GANs) and their superior ability in data generation compared to auto-regressive models. The models discussed in this section are referred throughout this study.

2.2 Recurrent neural networks

Deep Neural Networks (DNNs) are a class of machine learning models that are very powerful in performing parallel computations. These have demonstrated excellent performances on difficult tasks such as speech recognition [15, 16] and object recognition [17, 18, 19, 20]. However, despite this power, the inputs and the outputs for a DNN should have a fixed size limiting the application of DNNs. This is a significant drawback as most of the real world scenarios are variable-size

inputs.

Recurrent Neural Networks (RNNs) generalize feed-forward neural networks to support variable length sequences [21, 22]. Given an input sequence of $(x_1, x_2, ..., x_T)$, a standard RNN produced an output sequence of $(y_1, y_2, ..., y_T)$ using following equations [23].

$$h_t = sigmoid(W^{hx}x_t + W^{hh}h_{t-1}) \tag{2.1}$$

$$y_t = W^{yh}h_t \tag{2.2}$$

Here, W^{hx} and W^{hh} are weight matrices. We can see how the hidden state (h_t) are updated depending not only with the current input (x_t), but also with the previous hidden state (h_{t_1}) making RNNs ideal for modeling sequences. However, as we can see from the equations, the length of the output will be as same as the input length. This is a restriction for scenarios where the output length can be different to the input length. For example, in a machine translation task, the number of words in the output translation needs not to be same as the number of words in the input sentence.

The simplest approach used in the literature to support variable output lengths is the encoder-decoder architecture. Here, the encoder maps the input sentence to a fixed length vector. Then, the decoder generates the output based on the fixed-length vector produced by the encoder [24]. An RNN or a Convolutional Neural Network (CNN) can be used as the encoder [25, 26, 27, 28].

Equation 2.3 explains the function of the encoder when the encoder is an RNN; it keeps outputting hidden states until it reaches the end of the sentence. Here, f can be any type of an RNN (such as an LSTM [2]).

$$h_t = f(x_t, h_{t-1}) \tag{2.3}$$

After all hidden states are produced, i.e., when the end of the source sentence is reached, the context vector c is produced using the hidden states produced. This is demonstrated in equation 2.4. Here, q is a non-linear function. For an

instance, Sutskever et al. [23] used q as $q(h_1, ..., h_{T_x}) = h_{T_x}$.

$$c = q(h_1, ..., h_{T_x}), \tag{2.4}$$

The decoder then produces its output based on the context vector c here and previously generated words. This is explained in equation 2.5. Here, the for probability modeling, either an RNN [23] or a de-convolutional neural network [29] can be used.

$$p(y) = \prod_{t=1}^{T} p(y_t|\{y_1, ..., y_{t-1}\}, c) \tag{2.5}$$

Equation 2.6 shows how the above probability distribution is modeled when the decoder is an RNN. Here, s_t is the hidden state of the RNN.

$$p(y_t|\{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c) \tag{2.6}$$

Even though this setting supports different input and output lengths, it suffers from the difficulty to model long term dependencies [30, 31, 32, 33]. Long short-term memory networks (LSTMs) are known for their capability to learn long-term dependencies [32] better than RNNs, so using LSTMs instead of RNNs in the encoder-decoder architecture have been experimented to give better results [23].

RNNs and LSTMs are frequently used in a wide array of tasks such as semantic parsing [34], sentiment analysis [35], logical inference [36], and question answering [37]. However, it is a known fact that their memory is limited and they have trouble in modeling long range dependencies [38, 39, 40]. To increase the memory capacity of these recurrent models, we can increase the layer size. For example, the machine translation model developed by Sutskever et al. [41] uses two extremely large and deep LSTMs for encoding and decoding the sentences [42]. However, this is not possible if we do not have a substantial amount of data. In other words, if we have a small amount of data, increasing the network capacity would make the network prone to overfitting of data. Therefore, the applicability of these models on low resource domains is minimal.

2.3 Attention Mechanisms

The need to encode any given source sentence to a fixed-length vector can be identified as a main limitation with the above encoder-decoder based approaches [2]. Here, the encoder, which is a neural network, needs to be able to compress all the important information of the sentence to a fixed length vector. This compression would be difficult with long sentences especially for those that are longer than the sentences in the training dataset. This scenario is empirically proven by Cho et al. [43].

To alleviate this issue, Bahdanau et al. [2] introduced an enhancement to the encoder-decoder model. Here, when the decoder is about to produce a word, the decoder is capable of focussing only on the words of the source sentence which are relevant to producing the word. In other words, the decoder produces each word based on the context vectors associated with the relevant words of the source sentence and the words it previously generated (see equation 2.7).This contrasts with the original encoder-decoder architecture, where the decoder only has access to the compressed fixed-length output vector from the encoder (see equation 2.6). In equation 2.7, instead of having a constant context vector c as in equation 2.6, we have a dynamic context vector c_i, that is tailored to the current translation state.

$$p(y_i|\{y_1, ..., y_{i-1}\}, c) = g(y_{i-1}, s_i, c_i) \tag{2.7}$$

Bahdanau et al. [2] achieved this dynamic calculation of the context vector using an attention mechanism. Figure 2.1 explains this.

They use a bidirectional RNN as the encoder[1]. The context vector c_i for the i^{th} step is calculated by,

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.8}$$

The weight α_{ij} is calculated by,

[1]Bidirectional RNN is an RNN and a reversed RNN. This setting scans the source sentence from both ends so that the resulting hidden states have contexts from both ends of the sentence

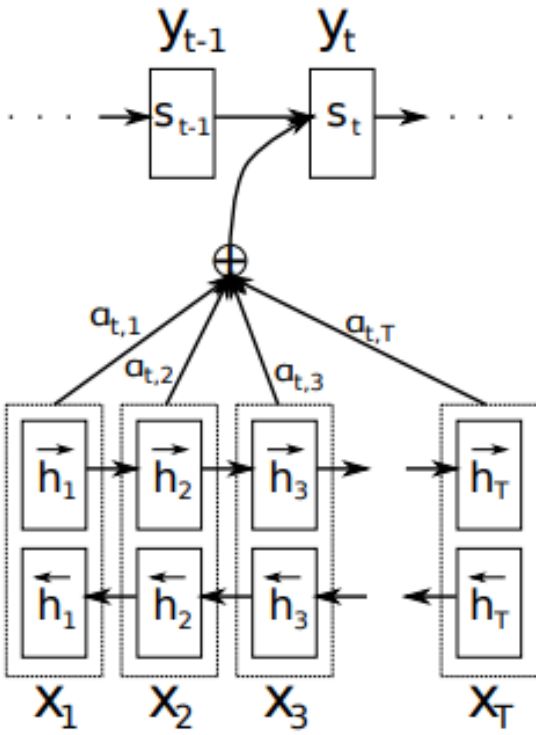


Figure 2.1: Attention mechanism introduced by Bahdanau et al. [2]. Image source: [2]

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.9}$$

where $e_{ij} = a(s_{i-1}, h_j)$. Here, a refers to the *alignment model* [2], that calculates the relevance of each hidden state h_j with the hidden state of the decoder s_{i-1}. This process of taking a weighted sum of all candidates is known as soft-attention. This is opposite to considering only a couple of candidates, which is known as hard-attention [44, 45]. Self-attention does not prevent gradients of the loss function passing through backpropagation [2]. Thus, this whole model is end-to-end trainable through backpropagation.

2.4 Memory Networks

Memory networks [39] were proposed as a means to query from multiple sentences. Memory networks work iteratively. Consider we have an N number of sentences as memories, with the memory for sentence i denoted as m_i. The memory network

first selects the most matching memory m_{o1} with the query x. Then it goes for a second iteration to select the most matching memory m_{o2}, but now, not only with the query x but also with the memory m_{o1}. This is explained in equations 2.10 and 2.11. We call these iterations as hops.

$$o_1 = \underset{i \in 1 \ldots \mathcal{N}}{\operatorname{argmax}} s_O(x, m_i) \tag{2.10}$$

$$o_2 = \underset{i \in 1 \ldots \mathcal{N}}{\operatorname{argmax}} s_O([x, m_{o1}], m_i) \tag{2.11}$$

This process is repeated for the configured number of hops (2 in this case). Finally, the most matching word ($\hat{a}$) from the vocabulary W is selected based on its similarity score with the query x and the selected memories (m_{o1} and m_{o2} in this case):

$$\hat{a} = \underset{w \in \mathcal{W}}{\operatorname{argmax}} s_R([x, m_{o1}, m_{o2}], w) \tag{2.12}$$

In the equations, s_O refers to *Output* module of the memory network, which makes the memory selection based on its matching with the information the module is having. s_R refers to the *Response* module, which is responsible for creating the response in the desired format with the input from the *Output* module and the query x.

Inside each iteration, memory networks use hard attention [41] for selecting the relevant memory; it selects the most matching memory using the *argmax* operation (see equations 2.10 and 2.11). Thus, memory networks cannot be trained end-to-end with standard backpropagation methods. So, to train memory networks without going for complex approaches such as variance reduction or reinforcement learning [41, 46], supporting sentences for each iteration are also included in the training data.

2.5 End-to-end Memory Networks

End-to-end Memory Networks [38] were introduced to eliminate the above limitation of being unable to train end-to-end. Here, instead of selecting the most

matching memory, this system follows a soft-attention mechanism where each memory is given a score based on its matching with the query. Equation 2.13 explains this; the dot product between each memory and the query x is computed, and then those scores are normalized (with the $Softmax$ operation) to get a probability distribution.

$$p_i = Softmax(x^T m_i) \tag{2.13}$$

Finally, we take the weighted sum of the memories according to their probabilities as the selected memory for the first iteration (equation 2.14).

$$o = \sum_i p_i c_i \tag{2.14}$$

After a hop, the question is updated with the information from the selected memory so that the selection of the next memory depends on the already selected memories (equation 2.15).

$$x_{k+1} = x_k + o_k \tag{2.15}$$

This process is considered a hop, and we repeat this process for the configured number of hops. Selecting the memory in terms of a weighted sum of all memories (soft attention) makes this system trainable end-to-end via standard backpropagation methods. Finally, the answer r is calculated by using a neural network (W). Input to W is the sum of the final output and the query, i.e., $o_h + x_h$ (equation 2.16).

$$\hat{a} = Softmax(W(o_h + x_h)) \tag{2.16}$$

Similar to memory networks, this network can produce only one-word answers. An RNN can be plugged instead of the neural network at the end (W) to generate answers with multiple words.

Three embedding matrices are used in an end-to-end memory network; one (embedding A) for embedding sentences to compute scores, one (embedding B) for embedding questions, and the remaining one (embedding C) for embedding the sentences to compute the weighted sum. Therefore, the trainable parameter

count under this setting roughly becomes, $3 * vocab\_size * embedding\_size$ for one hop. This count gets multiplied by the hop count resulting in a high parameter count, which is undesirable for a low-resource dataset. There are two types of modifications introduced by Sukhbaatar et al. [38] to reduce the parameter count; (i) Adjacent; eliminating embedding B altogether and using embedding A for embedding the question as well, sharing embeddings between hops (using embedding C_{k-1} for embedding A_k), and using final embedding C_k as W, and (ii) Layer-wise: having all input and output embeddings same throughout the hops i.e., $A_1 = A_2 = ... = A_k$ and $C_1 = C_2 = ... = C_k$.

2.6 Generative Adversarial Nets

Deep generative models are recently been given a lot of attention because of their ability to learn complex probability distributions from large unlabeled data. This is in contrast to traditional rule-based generative models that require human effort. Moreover, due to this requirement of human effort, maintainability and generalizability are difficult with rule-based solutions. Deep generative models alleviate these limitations and they have been proven to be more potential [47].

Deep Belief Nets (DBNs) [48] and Denoising Autoencoders (DAE) [47] are two of the first deep generative models introduce and they both learn to encode data to low-dimensional representations and generate the original data from the encoded representation. Both DBNs and DAEs learn a low dimensional representation (encoding) for each data instance and generate it from a decoding network. Variational autoencoder (VAE) use both deep learning and statistical inference to represent a data instance while using neural models for encoding [49]. All these models are trained by maximizing training data likelihood, which suffers from the difficulty of approximating many intractable probabilistic computations that arise in estimating maximum likelihood [50].

Goodfellow et al. [50] proposed Generative Adverserial Nets (GANs) as an alternative training methodology to generative models. Two models are trained here; a generative model G which that captures the data distribution, and a

discriminative model D which classifies between real and synthetic data from G. The training procedure is a minimax game between these two models where the training goal for G is to maximize the probability of D making a mistake. This framework alleviates the difficulty of maximum likelihood learning and has shown very good results in image generation [51].

Chapter 3

LITERATURE SURVEY

3.1 Automated solving and assessment of geometry problems

GEOS is the first system to focus on solving geometry questions in an end-to-end manner [3]. First, this system builds a lexicon by extracting function keywords along with their valences from the training dataset. For example, if there is a rule like "line(AB)", GEOS would extract the word "line" to be a function with the valence 1. GEOS uses a hard-coded set of rules to identify entities. For instance, if there is a word with two capital letters, GEOS records it to be an entity. A concept is either an entity or a function keyword. During the training phase, GEOS first scans a given natural language sentence to get the set of concepts using the lexicon and the hard-coded set of rules for the entities. Then the relationships within those concepts are generated, again, with a rule-based approach. This approach creates a large number of facts in a brute force manner. After this, the generated facts are scored for their accuracy using a discriminative model (logistic regression). Finally, relation completion is carried out using a rule-based approach. After this step, the accompanying diagram (if any) is used to filter the generated rules even more and to identify new facts. GEOS only focuses on solving SAT level MCQs in geometry. Hence, in the final step, the satisfiability of each choice is measured with the generated set of facts. As the answer, the choice that gets the highest matching probability is selected. Even though we can observe a large number of inaccurate rules produced at the end of text parsing, one reason this is not affecting GEOS is due to its limited scope to include only MCQs. Also, GEOS uses the diagram (if any) given with the question to filter the extracted set of facts even more. However, if this parser is used for problems where proofs are needed, not only might the wrong facts lead to a wrong proof, but also, the proving might even become impossible if some of the necessary facts

are missing in the extracted set.

GEOS uses a numerical approach (in checking the satisfiability with each choice to produce the answer) which is not intuitive for humans. GEOSV2 [8] was introduced by Sachan et al. modifying GEOS to use demonstrations for generating the answer instead of using this numerical approach, thus making a massive step towards interpretable geometry problem solvers. They have conducted a series of experiments to prove the user-friendliness and improved scores compared to GEOS. However, for parsing the question text, they have only replaced the brute-force parser in GEOS with a log-linear model that uses beam search to identify the concepts and the relations between them.

For extracting facts from the question text, both GEOS and GEOSV2 record high recalls (0.82 and 0.85 respectively), but comparatively low precision scores (0.57 and 0.59 respectively) i.e., these systems generate significant numbers of wrong facts. Also, these systems cannot be used for languages other than English as they depend heavily on prevailing NLP tools for the English language (such as dependency parsers and POS taggers), and hard-coded rules. Moreover, they generate a set of facts for a given single sentence, but cannot relate among multiple sentences.

3.2 Relation Extraction

Extracting entities [52, 53] and relations between them [54, 55] from unstructured text is a central task in information extraction [56]. Traditional approaches for relation extraction have entity extraction as a predecessor step [57, 58]. Here, entities are extracted first and then, the relations between those entities are discovered. As this is a pipeline approach, errors in recognizing the entities will get transferred to the relation extraction process.

Recently, there has been a surge of interest on end-to-end relation extraction. Here, entities and relations are jointly recognized [59, 60]. This approach prevents errors from entity recognition getting transferred to relation extraction. This approach also allows to model cross task dependencies. The neural model by Miwa

and Bansal [61] is one of the first models for end-to-end relation extraction. It used a biDirectional LSTM to learn hidden word representations and then used a tree-LSTM [62] to encode the output of a parser. Then they used these representations to make local decisions for entity and relation extractions. This model outperformed the best statistical model [63] with much improved results. This demonstrated the strength of neural models for end-to-end relation extraction.

However, the approach of taking local decisions in this model ignoring the structural dependencies between incremental decisions is opposite to the approach used by statistical models that use well-developed structured prediction methods [60, 63]. This way of taking local decisions has been observed to result *label bias*, which would prevent globally optimal structures receiving optimal scores from the model. Zhang et al. [64] addressed this issue by building a structural neural model based on globally optimized models for structure prediction [65, 66]. This model not only outperformed the existing systems, but also showed the effectiveness of global normalization. However, this model uses several LSTM structures. (Our experiments showed that this model overfits to our data even under its simplest settings.)

3.3 Text generation

There is a recent emergence in research in text generation [67, 68, 69]. Concerning text generation, we can classify the tasks into two categories; text generation in the supervised setting, and the unsupervised setting [70]. In the supervised setting, the goal is to generate a text similar to a target set. From the referenced examples above, image captioning, and machine translation fall under this approach. In the second setting, where the task is unsupervised, the aim is to generate samples that are following a probability distribution similar (or close) to the probability distribution of a given reference set. In simpler terms, the models are expected to generate samples that look like reference samples.

Text generation tasks such as the ones mentioned above have successfully been implemented using neural models [67, 68, 69]. Two main variants of these models

can be identified; models based on Maximum Likelihood Estimate (MLE) and Language Generative Adversarial Nets (Language GANs). In principle, MLE models generate text using the context of prior generated words. During the training phase, the next word of the sequence is predicted based on the ground truth words, and in the inference phase, the prediction is done using already predicted words [71, 12]. When it comes to GANs [50], they have been able to make significant advances in the generation of synthetic data similar to real data. Two neural networks are used here; the generator is responsible for generating realistic looking data, while the discriminator's responsibility is to differentiate between synthetic and real data accurately. The gradient of the training error from the discriminator is used to train the generator.

There are multiple identified issues with the MLE approach. Firstly, there is no appropriate metric to evaluate the output of these models [12]. As discussed above, MLE models use ground truth words as the context in the training phase while predicted words are used as the context during the inference phase. This discrepancy makes MLE models suffer from exposure bias i.e., model having to operate on unseen contexts during inferencing [12, 71]. Scheduled Sampling introduced by Bengio et al. [72] to address this problem was proved fundamentally inconsistent by Huszar [71].

The original setting of GANs works well when they are operating on continuous data (image pixel values). However, when it comes to text generation, GANs have to deal with discrete tokens (sequence words) that are non-differentiable. Therefore the usage of GANs is rather challenging due to the difficulty of back-propagation through these random discrete variables [73].

Recent attempts to face this challenge could be classified into two categories. Reinforcement Learning(RL) based approaches model this task as a sequential decision-making problem. GANs in this category use policy gradient techniques for optimization. SeqGAN by Yu et al. [10] is an example of this approach. One of the main drawbacks of SeqGAN is that the generator only gets the reward at the end of generating the whole sequence, thus making it difficult for the generator to sufficiently learn the distribution [12]. MaliGAN [73] uses a modified

optimization algorithm to reduce the high variance caused by the original form. RankGAN[74] replaces the original discriminator, which is a binary classifier, with a ranking to reduce the gradient vanishing problem experienced with the binary form. This replacement is also a solution for the information given by the binary classifier being inadequate. The discriminator being a binary classifier also contributes to mode collapsing [12]. Mode collapsing refers to the problem of a model generating samples only from a limited area of the latent space i.e., the samples being less in diversity with each other. LeakGAN [12] was proposed to mitigate the instability issues faced during the training phase under the standard RL approach. Here, the generator is given access to the feature representation learned by the discriminator. This specifically improves the generation of long text, as this brings more information to the generator network compared to the single binary signal in most of the previous GANs. Despite the performance enhancements promised by the above GANs that follow the RL approach, still, the high variance gradient they result makes the optimization challenging [11, 70, 75]. RL-free approach adheres to the original approach of GANs without incorporating ideas from RL. This approach does not yield gradients with high variance, so the GANs here are more stable and easier to train compared to the first category. TextGAN [11], GSGAN [13], and FMGAN [70] have adopted this approach and have generally reported better results compared to GANs in the previous approach.

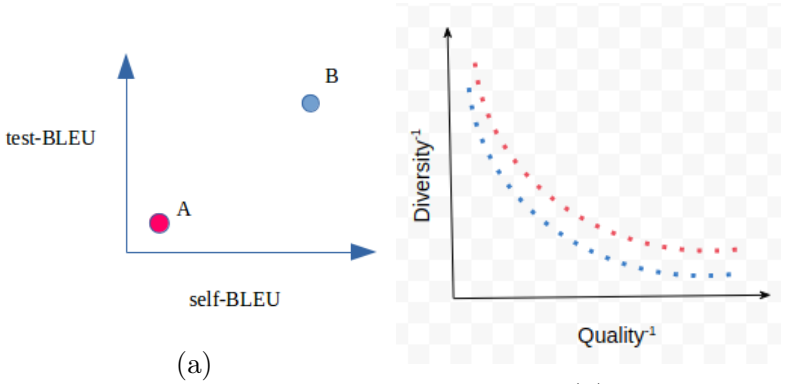
3.4 Evaluation of text generation models

When a generative model is trained, it attempts to learn a probability distribution that is similar to the probability distribution of the training dataset. Therefore, the perfect measure would be to measure the distance between the two probability distributions, which is called the *estimation error*. However, this is not practically feasible as we cannot use the total latent space to generate samples, and the reference set itself might not be fully representing its distribution. Therefore, some other metrics have been adopted, which are more feasible in a

practical setting. Test-BLEU is one such score that measures the similarity between two sets of text. It scores similar n-grams and their frequency. Therefore, this reflects sample quality. Most GANs (such as RankGAN [74], MaliGAN [73], TextGAN [11], LeakGAN [12]) focus only on sample quality in their performance comparisons [14]. This method is severely flawed that if a GAN generates a single quality sentence repeatedly, it will be able to get a perfect score [14]. Moreover, mode collapsing is a known issue in GANs, so they would anyways be biased towards generating sentences with less diversity.

Self-BLEU, a score to measure the diversity of a generated set of samples was proposed by Zhu et al. [76]. Here, the sample is analyzed against itself, so a low score(low similarity) means the sentences in the sample are diverse. Even though this makes the problem above solved, this makes the comparison of GANs difficult. For an instance, consider figure 3.1a where the two markers represent the scores of two GANs named A and B. Here, GAN A has a better diversity (low self-BLEU), but a low quality (low test-BLEU). GAN B has the opposite; a better quality, but a poor diversity. Hence this graph is not sufficient to decide on the better performing GAN. For evaluating these models, Texygen [77] has been proposed as a benchmark framework. This framework scores quality (using metrics such as test-BLEU, EmbSim) and diversity (using metrics such as self-BLEU) *separately*, so suffers from the same issue.

The entropy of a model is a value that decides the diversity of the results it produces. A low entropy would force a model to produce results with high probability, thereby less diverse, but high-quality results. A model with a high entropy would generate more diverse, but lesser quality samples. Boltzmann temperature [78] refers to the parameter that is used to change a model's entropy. I.e., high-temperature values make a model's entropy high, which results in generating data with a high diversity and, similarly, low-temperature values make the model stick to generating quality samples, which would not be much diverse from the reference set. Caccia et al. [14] proposed a novel approach for evaluating text generation models based on this scenario. This involves moving a model across a set of temperature values so that its performance on both diversity and quality



(a) (b)

Figure 3.1: (a) Two examples NLG models named A and B plotted on a graph with the x-axis as self-BLEU and the y-axis as test-BLEU. (b) Two example temperature curves of two NLG models. The axes have the inverse metrics, so the lower is better in both axes. When it comes to the model with the blue curve, in any given quality, it has better diversity than the model with the red curve. Therefore, this graph is sufficient enough to claim that the model with the blue curve is better.

aspects could be assessed. If the diversity and quality of the generated samples at each temperature are plotted, on a plot with the x-axis as $quality^{-1}$ and y-axis as $diversity^{-1}$, we can get a curve. We call this the temperature curve of a model. Here, as both axes are inverted, the closer the curve to the origin, we can say the model performs better with both quality and diversity aspects of the generated samples. Hence, if a model's temperature curve is situated below the temperature curve of another model (refer to figure 3.1b), we can say the first model is better than the second. This allows us to use the area under the curve as a performance measure [14].

Chapter 4

Deep semantic parsing of geometry problems

4.1 Task Definition

Table 4.1 illustrates the intended use of our model. Our model should initially produce unary rules (rules with one child), binary rules (rules with two children), and, finally, completed relations (merging unary and binary rules) for a given set of sentences. Also, it should be capable of extracting facts related to a given entity as shown under *On-demand fact Extraction* section in the table.

Considering the requirements above, we model this task of deep semantic parsing of GWPs as a question answering (Q/A) task. Through this way of modeling, in addition to the capability to extract facts similar to the other existing systems, we also facilitate the solver to extract facts about interested entities during the deduction process. This effectively increases the recall of the system lessening the severity of the impact caused by imperfect recalls during the initial fact extraction phase. We term this feature as **on-demand fact extraction**.

We build on memory networks. First, a list of keywords from the training data is automatically extracted along with their valences. For example, if there are two rules like {"parallel AB CD", "line AB"}, words *parallel*, and *line*, along with 2 and 3 as their valences, are extracted respectively. Then the unary model is trained for unary rules by taking only the sentences that have unary rules. Similarly, a model for binary rules is trained. The unary model produces single word rules ("AB" as per the example above), while the binary model produces rules with two words ("AB", "CD").

4.2 Limitations of existing memory networks

End-to-end memory networks, as discussed in the literature review, store a given sentence as the sum of the embedding vectors of its words resulting in a sentence

Table 4.1: Task Definition

Sentences	
Line AB is parallel to line CD AB is a chord of the circle with center O	
Initial fact Extraction	
Unary rules	Line(AB), Line(CD), Circle(O)
Binary rules	Parallel(AB, CD), IsChordOf(AB, O)
Relation Completion	
Parallel(Line(AB), Line(CD)) IsChordOf(Line(AB), Circle(O)))	
On-demand fact Extraction	
AB→ {Line(AB), Chord(AB)} CD→ Line(CD) O→ Circle(O)	

representation of the word embedding size. Even though this has not caused a significant loss of information with the experiments carried out by Sukhbaatar et al. [38], in our experiments, however, we saw this representation was not capable of retaining the information needed enough to produce correct facts. Even though increasing the network capacity will increase the model's ability to retain data, it is not viable in our task due to the small dataset size. On the other hand, if we can store only the important memories (instead of storing all the memories), we will not need to compress the information as much as above. This is our motivation behind coming up with a model that can selectively store only the essential data within a limited capacity.

4.3 Model Formulation

In order to store sentences in a less compressed manner within a limited capacity, we propose storing only the important sentences. Thus, we first need to select the sentences that are relevant to the query at hand. Taking x as the query and m_i as the i^{th} sentence, the probability for the relevance of sentence i to the query x is calculated by:

$$p_i = Sigmoid(x^T m_i) \tag{4.1}$$

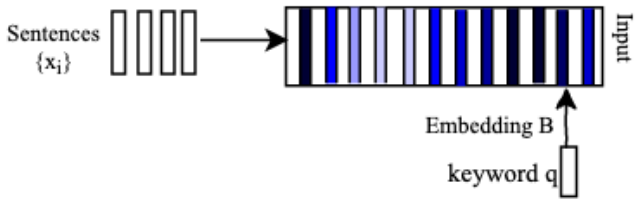


Figure 4.1: Architecture of the sentence selection model. The lighter the color, the higher the probability. This design relates to the single hop scenario, which is sufficient for the task we focus on. However, for tasks needing to handle inter-dependencies between sentences (such as coreference resolution), we can adopt the same word-level model (figure 4.2) with having sentences instead of words.

Figure 4.1 shows an example for the resulting probability distribution. The lighter colored sentences are sentences with higher probabilities. We do not concern about handling inter-sentence dependencies (such as coreference resolution) in this research. So we are not going for a multi-hops approach for selecting the relevant sentences. As a hyper-parameter of the model, we maintain a threshold for the lowest probability a sentence should get to be considered as a relevant sentence; the lower the threshold, more sentences get to be considered for fact extraction, which might result in more facts thus reducing the precision while increasing the recall. The opposite will happen if a higher threshold is selected. This process is called hard attention, and using this makes the network incapable of being trained using standard backpropagation methods [41]. However, to make the computations simple and to retain the interpretability, we give up end-to-end trainable capability.

After selecting the relevant sentences, each of the selected sentences is fed to the **word-level memory network**. Here, the model attends each selected sentence in the word level. Figure 4.2 refers to this model in the single hop scenario.

First, the words are embedded with the embedding matrix A, and the query is embedded using the embedding matrix B. Then the dot product is calculated between each word embedding vector and the query. These scores are converted to a probability distribution using the $Softmax$ operation (equation 4.2).

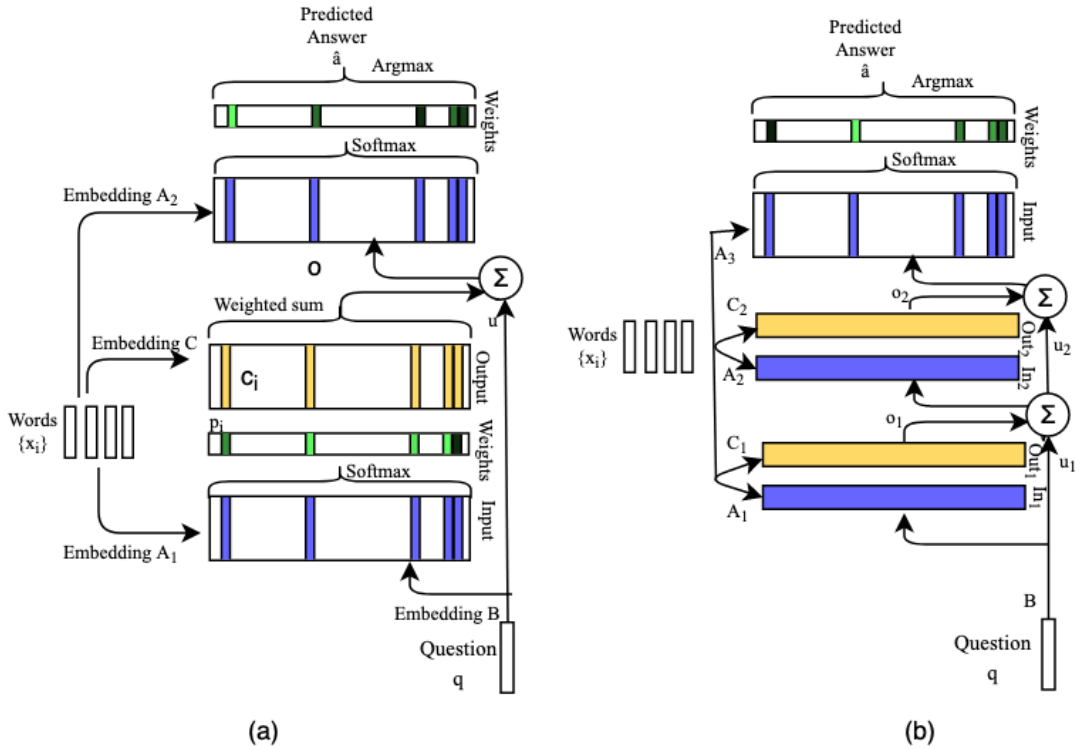


Figure 4.2: Architecture of the Word-level memory network

$$p_i = Softmax(x^T m_i) \tag{4.2}$$

After that, the words are again embedded using the embedding matrix C, and then the sum of these embedding vectors is calculated. When taking this sum, each vector c_i is weighted by its probability (p_i) calculated above (equation 4.3).

$$o = \sum_i p_i c_i \tag{4.3}$$

Now the query embedding is updated with this sum so that the query for the next hop contains the information from this hop (equation 4.4).

$$x_{k+1} = x_k + o_k \tag{4.4}$$

In the single hop scenario (figure 4.2a), the words are embedded using a second input embedding matrix (A_2), and the dot products are calculated with the updated query. Then, these dot product values are converted to a probability

distribution using the $Softmax$ operation. This process is similar to the process described above (equation 4.2). Now, as the answer, the word with the highest probability is selected (equation 4.5).

$$\hat{a} = Argmax(Softmax(x_{k+1}^{T} m_i)) \tag{4.5}$$

In the multi-hop scenario (Figure 4.2b), equations from 4.2- 4.4 are executed in a loop for the given number of hops. After each hop, the query is updated with the weighted sum of the word embeddings. This process of updating the query allows the forward-passing of information. Finally, similar to the single hop scenario, the answer is selected based on the probabilities calculated in the final hop.

4.3.1 Focus controlling (FC) with position encoding

If we consider the sentence “In the given figure, line AB is parallel to line CD”, it contains two rules for the same keyword “line”. If we query the sentence just with “line”, we will not be able to get both rules. We overcome this limitation of being unable to retrieve multiple rules for the same keyword by introducing a mechanism to "tell" the network where to focus. We do this by a customized **position encoding** (PE) scheme. This scheme also serves the purpose of giving the model a sense of the order of the words. We build our scheme based on the PE scheme introduced by Sukhbaatar et al. [38]. We first extend the embedding matrices with a size of $2 * max\_len$, where max_len refers to the maximum sentence length in the training dataset. After that, instead of the memory m_i being simply the embedding of w_i, we modify the memory m_i to be the $embedding(w_i) + embedding(max\_len + d_i)$, where d_i refers to the distance between w_i and the matched keyword in the sentence. Words before the query word get negative values for d_i, whereas words after the query word get positive values for d_i. According to this scheme, the memory in the location of the query always gets added to the same vector; $embedding(max\_len)$. So does the words around the query word. Hence, we can expect the model to focus more on

memories near the query word.

4.3.2 Unary Rule Extraction

This task only needs single word answers. Therefore, we use the model in Figure 4.2b as it is.

4.3.3 Binary Rule Extraction

Here, we focus on rules such as "parallel AB CD". Unlike extracting unary rules, not only do we have to extract two words, but also the second word depends on the first word retrieved. Due to this dependency, we cannot model this problem as a multi-class classification problem. Also, due to the reasons explained earlier, introducing an RNN will bring in a large number of trainable parameters. This large count is undesirable due to the limited dataset size. Therefore, inspired by the original architecture of memory networks [39], we come up with a layer-wise retrieving mechanism; assuming we have k number of hops and $layer_k$ as the final layer, we retrieve the first literal ("AB" in this case) from $layer_{k-1}$ and the second literal ("CD") from $layer_k$. We define the loss function as the summation of individual losses (categorical cross-entropies) from the two layers;

$$Loss = \sum_{i}^{l} (t_{i_k} log(s_{i_k}) \ + t_{i_{k-1}} log(s_{i_{k-1}}) \ ), i = 1, 2...l \tag{4.6}$$

Here, l refers to the memory size (or the length of the sentence). t_{i_k} refers to the ground truth value for the i^{th} location for k^{th} layer. Usually (if label smoothing or any such technique is not used), t_{i_k} is 1 for the location of the second literal ("CD") and 0 for the other locations. Similarly, $t_{i_{k-1}}$ is 1 for the location of the first literal ("AB") and and 0 for the other locations. s_{i_k} refers to the probability computed by the model for i^{th} location for k^{th} layer.

4.3.4 On-demand Fact Extraction

This task makes the model interactive by facilitating to extract properties of entities **on-demand**. Here, a user can query the system to retrieve information

based on their needs, which can be dynamic. For example, during a proof, a solver would change its focus on entities as it progresses on deducing new relationships among entities. This feature also alleviates the problem of having an imperfect recall. A solver can use this feature if it wants to retrieve the rules that were missed in the tasks above. Existing systems do not provide this capability.

We model this task similar to unary rule extraction. Here, we query with the interested entity. For example, if we consider the sentence "AB is a tangent to circle O", and if we query the sentence with "AB", the system will produce "tangent". Through FC, as described above, we are capable of retrieving multiple rules for a single entity.

4.4 Experiments

Experiments of GEOSV2 have been carried on a dataset that is around 7.5 times larger than the dataset used in GEOS. This dataset is not publicly available. So we are not able to compare the scores of our model with the scores recorded by GEOSV2. Thus, we compare the scores of our model with the scores of GEOS and GEOSV2 when all of them are trained on the dataset provided by Seo et al [3]. This the dataset used in GEOS. Table 4.2 shows the details of this dataset. We use the training dataset there as our training dataset and the practice dataset as our evaluation dataset,

For the tasks of unary and binary rule extractions, we use the keyword list extracted from the training dataset. For example, if there are two rules like "parallel AB CD", "line AB", the words "parallel", and "line" are extracted along with two and three as their valences respectively.

4.4.1 Unary Rule Extraction

Keywords with valence 1 are used for this task. First, the given sentence is scanned for keywords. If a keyword makes a match with a word, the model is queried with the keyword. Here we use 30 as the batch size, 3 hops, 70 epochs, 50 as the embedding size, and layer-wise weight sharing ($A_1 = A_2 = ... = A_k$

Table 4.2: Statistics of the dataset. Introduced by Seo et al. [3]

	Total	Training	Practice	Official
Questions	186	67	64	55
Sentences	326	121	110	105
Words	4343	1435	1310	1598
Literals	577	176	189	212
Binary relations	337	110	108	119
Unary relations	437	141	150	146

Table 4.3: Precision, Recall, and F1 score for the task of unary rule extraction

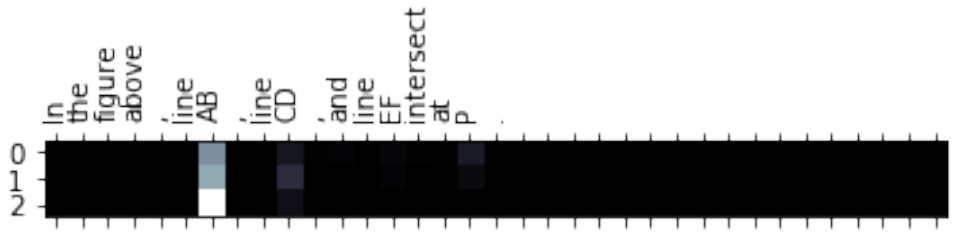
	P	R	F1
End-to-end MN	0.51	0.25	0.33
Two-step MN without FC	0.52	0.42	0.46
Two-step MN with dynamic FC	0.55	0.58	0.56
Two-step MN with fixed FC	**0.68**	**0.72**	**0.70**

and $C_1 = C_2 = ... = C_k$).

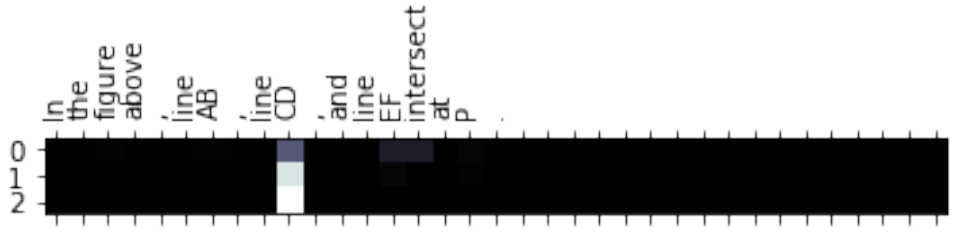
Table 4.3 indicates the results of this task. We can see a significant improvement in the F1-score when it comes to two-step memory networks from end-to-end memory networks. This improvement validates our design for this task.

We handle multiple occurrences of the same keyword through FC. If not for FC, the model will produce the same rule for each occurrence of the keyword. Figure 4.3 demonstrates how multiple rule extraction happens for the same keyword. In figure 4.3a, the query is focused on the first occurrence of the word "line". Second and third occurrences are focused in Figures 4.3b and 4.3c respectively. These figures demonstrate how probability distributions change based on focused locations.

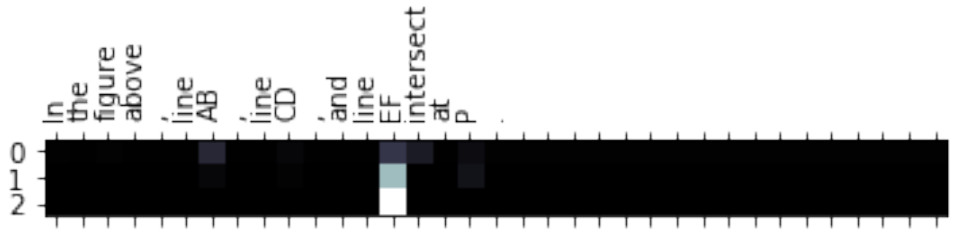
Dynamic versus fixed FC: Dynamic FC refers to what we have discussed above; changing the position encoding based on the location of the keyword. Consider the sentence "In the figure below, line AB is parallel to line CD". We use {"line", 5} and {"line", 10} for querying under this setting. Fixed FC refers to providing the network the first position of the keyword despite the occurrence we are focusing on i,e, for both occurrences, we use the 5 as the query word position. With fixed FC, a significant number of duplicate facts are produced. However,



(a) Extracting the rule "line AB"



(b) Extracting the rule "line CD"



(c) Extracting the rule "line EF"

Figure 4.3: Querying the sentence "In the figure above, line AB, line CD, and line EF intersect at P." with the keyword "line".

the duplicates are removed for every task as a post-processing step. Scores are calculated after removing the duplicates.

Interestingly, when it comes to results, we can see that dynamic FC has lower scores compared to fixed FC (3^{rd} and 4^{th} rows of table 4.3). We can see that the network gives more confident results when trained without having to concern multiple query word locations. The disadvantage caused by being unable to produce multiple rules for the same keyword can be seen to be overridden by the advantage of being able to ignore the query word position during training. As expected, having FC in either setting above is better than having no FC (2^{nd} and 3^{rd} rows of table 4.3)

Table 4.4: Precision, Recall, and F1 score for the task of binary rule extraction

	P	R	F1
End-to-end MN	**0.83**	0.19	0.30
Two-step MN without FC	0.36	0.60	0.45
Two-step MN with fixed FC	0.49	**0.62**	**0.55**

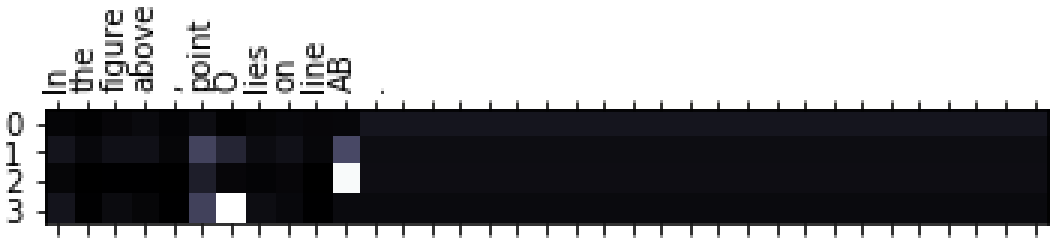


Figure 4.4: Binary rule extraction for the keyword "lies" from the sentence "In the figure above , point O lies on line AB.". Here we can see how the the two literals ("O" and "AB") are with high probabilities in the last two layers.

4.4.2 Binary Rule Extraction

First, the keywords with valence 2 are filtered for this task. Then, the same process of scanning followed in unary rule extraction above is followed. Here we use 10 as the batch size, 4 hops, 100 epochs, 100 as the embedding size, and layer-wise weight sharing ($A_1 = A_2 = ... = A_k$ and $C_1 = C_2 = ... = C_k$). We did not use dynamic FC due to the complexity and low results even for the fixed FC here (table 4.4). However, we can see a significant improvement in the f1-score with the two-step memory network (two-step MN) and further improvement with FC. So, we can claim that the improvements we present for the original architecture of memory networks are valid for this task as well. Figure 4.4 shows an example for this task.

4.4.3 On-demand Fact Extraction

Dataset Preparation

For training the model for this task, we modified the dataset of GEOS. We removed the unary rules that were not focusing on entities (rules such as "six triangles"). Then we manually annotated rules for entities that did not have unary

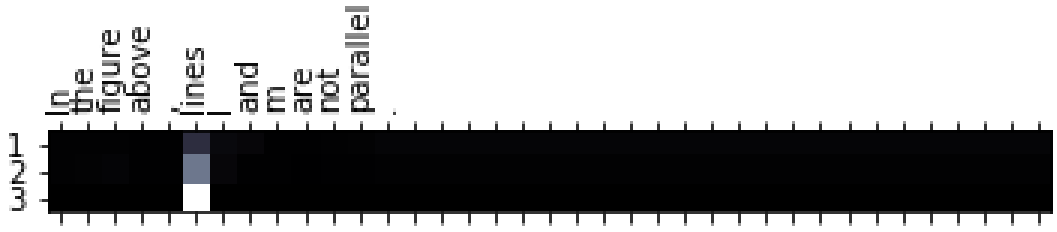


Figure 4.5: Fact extraction for the entity "m" from "In the figure above, lines l and m are not parallel." Lighter colors indicate higher probabilities

rules. For example, the sentence "Lines AB and BC are parallel" contained a unary rule only for "AB" as "Lines AB". So we added another rule as "Lines BC". This step of adding rules was needed to prevent the misclassification of correct rules produced by the model. Figure 4.5 demonstrates an example for this scenario. There, even though the model extracts the rule "lines m", there is no such rule in the original dataset. Finally, we reversed the modified ruleset so that the entity would be the querying word. For example, for the sentence given above, the model would be queried with "AB" to get the rule "AB lines". In the end, we got a total of 319 rules.

Training and Evaluation

We train the model with this dataset. We use 10 as the batch size, 3 hops, 100 epochs, 50 as the embedding size, and layer-wise weight sharing as the hyper-parameters for this task. For the testing phase, we assume all single characters (both lower and upper case), and words with all capital letters as entities. Using a regex expression, we first scan the sentence to retrieve all the candidates fitting the above assumption. Then we query the sentence with each of those candidates. It is important to notice that we use this regex rule only for evaluating the system under this task. We do not need this rule for training the system and inferencing afterwards. Figure 4.6 shows how multiple hops help here. We expect the model to produce every rule related to the entity it is queried on. So we cannot use fixed FC here. Table 4.5 shows our results under this task. Similar to the above tasks, we can see substantial improvements in the results with the improvements

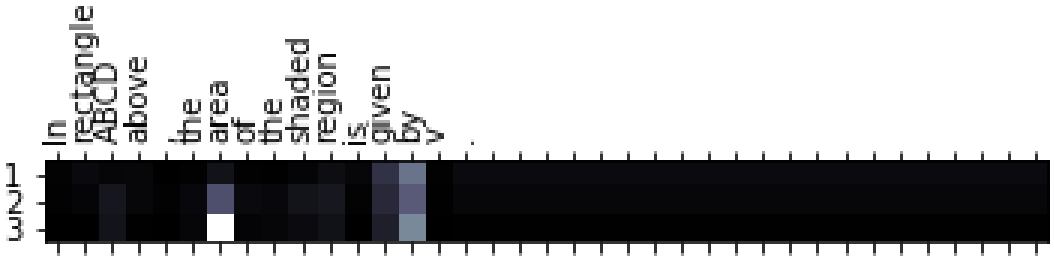


Figure 4.6: Fact extraction for the entity "v" from "In rectangle ABCD above, the area of the shaded region is given by v". Here, we can see how the answer "area" is refined over hops. If there were only a single hop, the answer would have been the word "by" (word with the lightest color in row 1) which is incorrect.

Table 4.5: Precision, Recall, and F1 score for the task of on-demand fact extraction

	P	R	F1
End-to-end MN	0.49	0.20	0.29
Two-step MN without FC	0.49	0.56	0.52
Two-step MN with dynamic FC	**0.73**	**0.8**	**0.76**

we introduce.

4.4.4 Relation Completion

In this section, we see whether our system outperforms the existing systems. As already described, the problem of parsing MWPs in geometry is only explored recently [3], and we have two systems named GEOS [3] and GEOSV2 [8] for this task. Experiments of GEOSV2 have been carried on a dataset larger than the dataset used in GEOS. This dataset is not publicly available. So we are not able to compare the scores of our model with the scores recorded by GEOSV2. Therefore, we train GEOSV2 on the public dataset introduced by Seo et al. [3]. We use the training dataset there as our training dataset and the practice dataset as our evaluation dataset,

The results are indicated in table 4.6.

Compared to GEOS, we have an increment of 25% for precision even though we have a drop of F1-score by 3%. Compared to GEOSV2, we have a precision increment of 20% while we record a drop of 3% with F1-score. Both GEOS and

Table 4.6: Precision, Recall, and F1 score for relation completion

	P	R	F1
GEOS	0.57	0.82	0.67
GEOSV2	0.59	**0.83**	**0.69**
Two-step MN	**0.71**	0.60	0.65

GEOSV2 have low precision scores despite the high recalls, which occurs when a system produces a lot of false positives.

4.5 Discussion

Existing systems have better F1-scores and recalls than our system does. However, we record a higher precision than these systems and a close f1-score. These systems have low precision scores despite their high recall scores, which occurs when a system produces a lot of false positives. So we can claim that the rules generated by our system are more reliable than the rules created by GEOS. Also, our system can generate on-demand rules which alleviates the problem of having a low recall. Most importantly, these systems are heavily dependent on NLP tools and hardcoded parse rules. Our system depends on neither, yet achieve comparable results. Since low-resource languages do not have NLP tools such as POS taggers and dependency parsers, these systems cannot operate in those languages. In contrast, our system is compatible with any language.

The success of applying two-step memory networks on this low resource dataset without feeding any domain-dependent or language-dependent input lead the way to many similar applications. The significance of our contribution lies in the fact that we applied deep learning to a low resource dataset and achieved results comparable with rule-based and language dependent systems.

As for the limitations, one main limitation of our system compared to end-to-end memory networks is the inability to train end-to-end. In other words, supporting sentences need to be annotated in the training dataset, whereas it is not needed for end-to-end memory networks. Also, in using two memory networks (sentence-level and word-level), we increase the number of training parameters

compared to using a single memory network. If this increment is undesirable for a task, one option is to parse individual sentences, i.e., use only the word-level memory network. Also, the parameter count can be reduced using parameter sharing techniques described above, and by decreasing the hop count.

Chapter 5

TEXT GENERATION ON LOW-RESOURCE DOMAINS

Generating meaningful and coherent sentences has many applications in natural language processing. Hence, there is a growing interest in this area of research [63, 79, 80, 81, 82, 83, 84]. The general idea is to estimate a distribution over sentences from a corpus, then use it to sample realistic-looking sentences. This task is important because it enables generation of novel sentences that preserve the semantic and syntactic properties of real-world sentences, while being potentially different from any of the examples used to estimate the model. For instance, in the context of dialog generation, it is desirable to generate answers that are more diverse and less generic (Li et al., 2016).

Recently, neural models have successfully been employed to generate text in multiple contexts such as image captioning [67], machine translation [68], and poem generation [69]. We can classify these tasks into two categories; text generation in the supervised setting, and the unsupervised setting [70]. Under the supervised setting, the goal is to generate a text similar to a target set. From the referenced examples above, image captioning, and machine translation fall under this approach. Under the second setting, where the task is unsupervised, the aim is to generate samples that are following a probability distribution similar (or close) to the probability distribution of a given reference set. In simpler terms, under the unsupervised setting, the models are expected to generate samples that look like reference samples.

There are mainly two classes of the Natural Language Generative (NLG) models that fall under the supervised setting: models based on Maximum Likelihood Estimate (MLE), and language Generative Adversarial Nets (language GANs). Concerning language GANs, many different models have been proposed. These models use different kinds of techniques. To compare these models under similar settings, Texygen [77] has been introduced. Albeit the vast amount of exper-

iments carried on this framework extensively comparing these models, all the experiments reported by Texygen have been carried out on large corpora such as MS COCO captions [85] and EMNLP2017 WMT News [12].

However, the availability of large datasets would not hold in many domains. For example, the amount of data available in the GEOS [3] system for solving geometry questions is under 500 sentences. With the questions being complex and diverse, this low quantity inhibits any application of deep learning models for parsing these questions. On the other hand, expert knowledge is required to manually create such a dataset. Thus, mechanisms to automatically generate more questions from such seed corpora are imperative. Machine translation in low-resource languages is another such task where the developers have to resort to manual feature engineering due to lack of data sufficient enough to go for neural approaches.

5.1 Task Definition

Under this work, we analyze the applicability of the state-of-the-art NLG models on small seed corpora. We evaluate the vanilla MLE language model [77] and four language GANs (SeqGAN [10], TextGAN [11], LeakGAN [12], and GSGAN [13]) using small datasets extracted from the datasets used in Texygen, most importantly, with an eye to reducing the variance. We also analyze the applicability of these NLG models in terms of the inherent properties of the reference datasets such as the average sentence length and language complexity.

5.2 Experiments

5.2.1 Enhanced metric for evaluating NLG models

As discuused in the literature review, if a model's temperature curve is situated below the temperature curve of another model, the first model can be claimed to be better than the second model. This allows using the area under the curve as a performance measure [14]. However, we argue that this setting is misleading

when the curves are are crossing each other. For example, consider a model that is repeatedly producing the same high quality sentence over and over again, and another model that produces a bit more diverse data with average quality. Here, we claim that this metric would rank the first model above the second model owing to the first model's high contribution to the area with its very high quality values.

For calculating the area needed by the Temperature sweep method, we adopt summing the normalized areas under the curve with respect to x-axis and y-axis. As discussed above, these area values can be misleading. On the other hand, the areas under the x and y axes reflect the diversity and quality aspects respectively. So, by analysing these areas separately, we can identify the models which gives acceptable performances under each aspect. Thus, we can filter those models and use the final graph (which is referred under the Temperature Sweep method) to rank only the filtered models. Under this approach, manipulating the final area by a model that is extremely good in one aspect while being worse with respect to the other aspect is impossible.

5.2.2 Experiment Setup

Here, we aim to analyze the applicability of NLG models on low-resourced domains. We also aim to analyze the accuracy of the Temperature Sweep method in evaluating NLG models by individually considering the impact from individual scores.

We use Texygen to run our experiments. We experimented with two datasets: MS COCO captions [85], and EMNLP2017 WMT News [12]. Table 5.1 gives a summary of the statistics of these two datasets. We make our low-resource by extracting partitions from these two datatsets. We extract multiple partitions here to reduce the variance which occurs when extracting a small dataset from a large dataset. Our method is described below.

We experiment for dataset sizes ranging from 200 to 1000 in 200 step sizes to cater to our need to examine the behaviours of NLG models on low-resource

domains. For each data size, we increase the pre-epoch count (the number of epochs the generator is pretrained for) by 10 starting from 10, and the adversarial epoch count (the number of epochs the network is trained adversary for) by 10 starting from 20. These values were selected based on the values used by Texygen and dataset size. To deal with the large variance that occurs between small datasets extracted from a big dataset, we experiment with three different partitions for a given large dataset and calculate the average. First, we separate three random datasets of size 1000 from the training dataset as *base training sets*. These three are used to get the data needed for experiments. For instance, if we are evaluating when the dataset size is 400, we take three sets of data sized 400 from the three base training sets. The same process is followed with the test sets, but with altered numbers. We take 5 sets of size 200 as testing sets. To analyze the trend of performance with increasing dataset sizes, we kept test dataset size (200) constant while varying training set sizes. So, to get its quality and the diversity scores of a generated set of samples, it would be scored against each of the 5 test sample sets, and then averaged.

After training a model with a training set, we generate samples of size 200 for a set of temperatures ranging from 0.001 to 2.0 with 0.5 step sizes. Based on work by Caccia et al. [14], the temperature value could be varied starting from zero to positive integers, and normally, the syntax breaks down losing the coherence of a sentence when the value is increased above 1.0. Therefore, we took 0.001 as the value equivalent to zero (to reduce overflowing and underflowing issues due to zero) and 2.0 as the temperature on the end. Compared to each test set, we calculate the testBLEU and selfBLEU scores with the n-gram count 3. These scores are averaged over the test sets.

Then this average score for each temperature is again averaged over the scores from the other two training sets. This is done for sizes ranging from 200. At the end of this process, we get a testBLEU score and a selfBLEU score per temperature per size. Then, for each size, we can draw a graph taking (1−testBLEU) as the x-axis and the selfBLEU as the y-axis. As the performance is directly proportional to the testBLEU and inversely proportional to the selfBLEU, the

area under this temperature curve is an accurate indicator [14]. Therefore, for each size, we need to calculate the area under its temperature curve. We divide this task into several steps: (*i*) compute the area under the curve w.r.t. the x-axis (this area depends on the selfBLEU score and independent on testBLEU), (*ii*) compute the area under the curve w.r.t. y-axis (this area depends on the 1−testBLEU score and independent on selfBLEU), (*iii*) normalize the area values to be between 0 and 1 so that the impacts from both quality and diversity aspects are equal, and (*iv*) get the total area as the sum of the areas calculated in the two steps above and normalize. The area calculated under step (*i*) relates to the diversity of the generated sample. The lesser this value, the more diverse is the sample. Similarly, the area calculated under step (*ii*) relates to the quality of the generated sample. The lesser this value, the more quality is the sample. The sum of these two values could be taken as the performance indicator for a given training set size for a given GAN. We take these area values to plot the final result graph. Algorithm 1 explains this.

5.2.3 Experiments with MS COCO captions

COCO captions [85] dataset contains a set of image captions that are smaller in length compared to EMNLP News dataset (refer Table 5.1).

Figure 5.1a indicates the area (normalized) of the temperature curve w.r.t (selfBLEU) axis. This graph relates to the quality of the generated data. So, lower the area, the better the model at producing quality data. Here, we can see GSGAN and TextGAN to have the largest values for all data sizes compared to other models. This implies that they are the worst performing GANs concerning quality. We can see other models in an improving trend (decreasing in area values, thus increasing in quality). Figure 5.1b indicates the area (normalized) of the temperature curve w.r.t 1-testBLEU axis. This graph relates to the diversity of the generated data. So, lower the area, the better the model at producing diverse data. We can see GSGAN going almost to zero when the dataset size is 1000. At this size, the generated samples from GSGAN have just spaces except

Algorithm 1: AREA generate area coordinates for different training dataset sizes for a given GAN

Input: Two finite sentence sets Tr and Ts for training and test set respectively, initSize, stepSize, and the finalSize of the dataset sizes evaluated

Output: (size, area) coordinates for the considered GAN

$Tr_1, Tr_2, Tr_3 \leftarrow$ 3 distinct sets of 1000 random rows from Tr without replacement

$Te_1, Te_2, Te_3, Te_4, Te_5 \leftarrow$ 5 distinct sets of 200 random rows from Te without replacement

$Tr \leftarrow \{Tr_1, Tr_2, Tr_3\}$

$Te \leftarrow \{Te_1, Te_2, Te_3, Te_4, Te_5\}$

$Temps \leftarrow \{0.0001, 0.5, 0.75, 1.0, 1.5, 1.75, 2.0\}$

$Areas \leftarrow \emptyset$

for $size \leftarrow initSize$ ***to*** $finalSize$ **do**

  for tr *in* Tr **do**

    get a random set of samples of size $size$ and train the model with it

    for $temp$ *in* $Temps$ **do**

      generate sample of $initSize$ under $temp$ temperature

      compare this with the 5 testing sets, average the testBLEU and selfBLEU scores

    `/* here we have a testBLEU score for each` $temp$ `in` $Temps$`. Same goes for selfBLEU */`

  average the testBLEU and selfBLEU scores over the 3 training sets

  `/* now we have a testBLEU score for each` $temp$ `in` $Temps$ `for the` $size$`. Same goes for selfBLEU */`

  nomalize both scores to be in [0,1]

  plot the scores in a graph with (1-testBLEU) as the x-axis and (selfBLEU) as the y-axis.

  calculate the area under the curve w.r.t. both axes

`/* now we have got an area value for each size w.r.t. both axes */`

normalize the area values to be in [0,1]

return *area scores*

Table 5.1: Summary statistics of the datasets used

Dataset	Train	Test	Vocabulary size	Avg. sentence length
MS COCO captions	120,000	10,000	27,842	11
EMNLP2017 WMT News	278,686	10,000	5,728	28

for a couple of words for the whole sample i.e. the generated set here has no more than 20 words. This extremely sparse spread of 20 or so words results in very low selfBLEU scores that the area has gone to zero or near zero. It can be seen that LeakGAN and seqGAN are on an increasing trend, while the MLE based model is diminishing on the increasing trend. TextGAN seems to follow the trend of the MLE based model.

Figure 5.2 is the normalized sum of the above discussed two areas; so the lower, the better. Interestingly, we can see how MLE model is at the bottom compared to the rest of the models. This implies it performs the best in quality and diversity combined. LeakGAN and SeqGAN can be seen performing alike as the size grows.

Figure 5.2 is the graph proposed by Caccia et al. [14] to evaluate the performance of NLG models alleviating the limitations of former metrics. Here, we can observe a negative slope for GSGAN, which is a characteristic of a good model. However, we saw that GSGAN is giving good diversity scores while being much worse in the scores related to the quality. Hence, the negative slope of GSGAN in this graph is misleading. Also, even though TextGAN seems to be performing well, we saw it has a low quality from Figure 5.1a. Figure 5.2 is also misleading due to the fact it shows as if TextGAN gets close to LeakGAN over the end, but TextGAN had the worst quality along with GSGAN as per the figure 5.1a. Therefore, in contrast to Caccia et al. [14], it is evident that the final graph alone with the area under the curve is not sufficient to evaluate the models. To alleviate this limitation, we suggest filtering the models that perform acceptably referring the graphs indicating the area scores under (1-testBLEU) and selfBLEU axes, and then ranking those filtered models from the final graph (graph of the sum of areas).

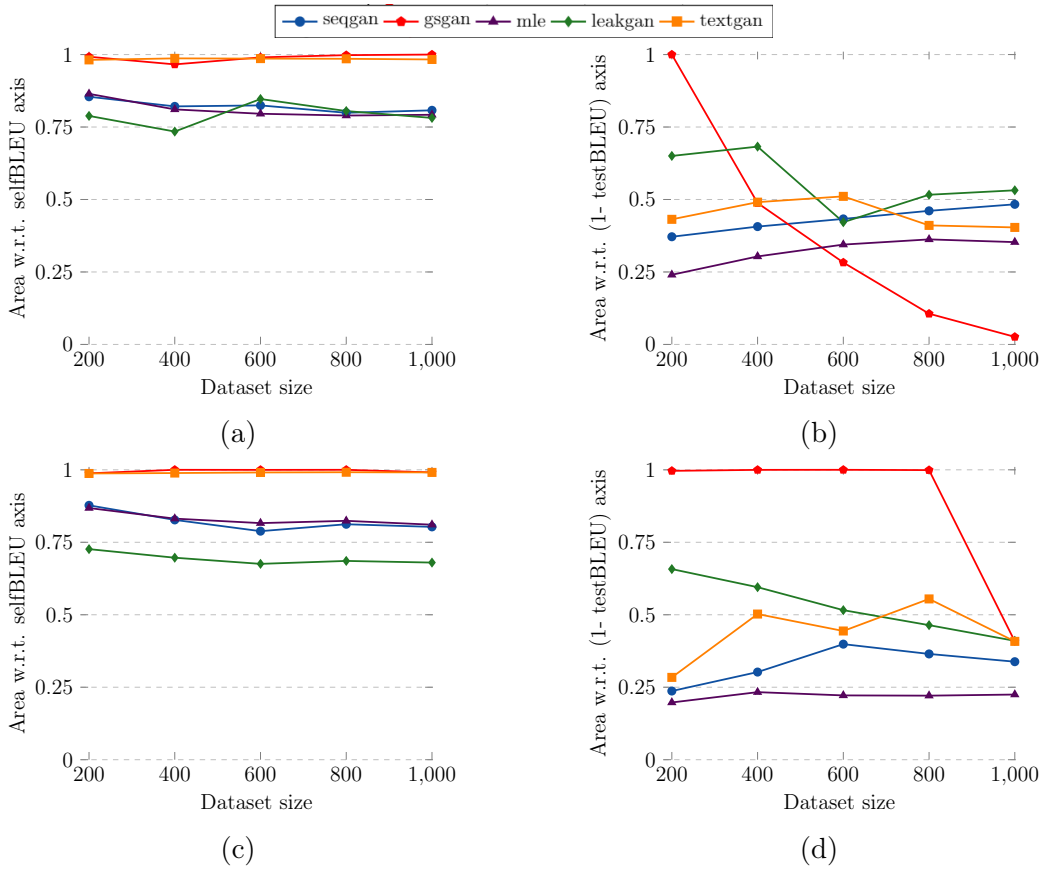


Figure 5.1: Area under the temperature curve of COCO dataset plotted with respect to (a) selfBLEU axis (b) (1− testBLEU axis) axis. Area under the temperature curve of EMNLP NEWS dataset plotted with respect to (c) selfBLEU axis , and (d) (1− testBLEU axis). The area w.r.t. selfBLEU axis is inversely proportional to the quality of the generated sample. Thus, higher values reflect poor quality samples, and vice versa. Similarly, the area w.r.t.(1− testBLEU) axis is inversely proportional to the diversity of the sample. Thus, higher values reflect samples with lesser diversities.

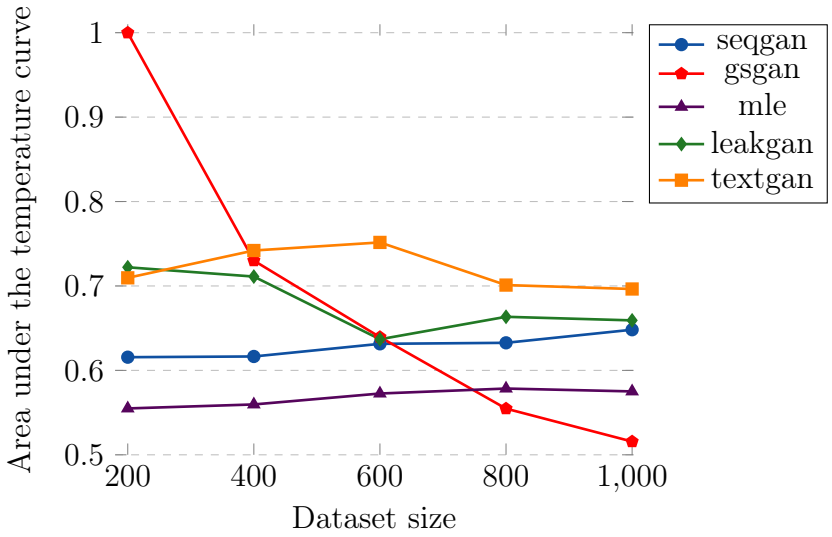


Figure 5.2: Experiment results on MS COCO dataset

5.2.4 Experiments with EMNLP2017WMT News Dataset

Sentences of this dataset are more complex and have a longer average length compared to the previous COCO dataset(Table 5.1). Similar to the graphs for COCO Dataset, figure 5.1c indicates the area of the temperature curve w.r.t selfBLEU axis. The values here corresponds to the quality. This graph is quite similar to the graph in figure 5.1a considering the worse behaviour shown by GSGAN and TextGAN. However, here, LeakGAN produces samples with the best quality for all the sizes while the MLE based model and SeqGAN perform quite similarly. Figure 5.1d indicates the area of the temperature curve w.r.t (1-testBLEU) axis. As explained above, the values in this graph relate to diversity; lower value means a higher diversity.

Interestingly, the MLE model seems to be winning here as well. GSGAN behaves quite different to the way it behaved during the experiments with COCO dataset. With COCO dataset, GSGAN generated a few words (around 20 words for the whole sample) with a lot of spaces. Here, it generates a small set of words repetitively. This makes its diversity score low. This happens when the dataset size is low. As the dataset size grows, GSGAN resorts to its normal behavior of generating a few words with a lot of spaces. This can be seen from the graph in figure 5.1d as well. This graph relates to the diversity of the generated sample.

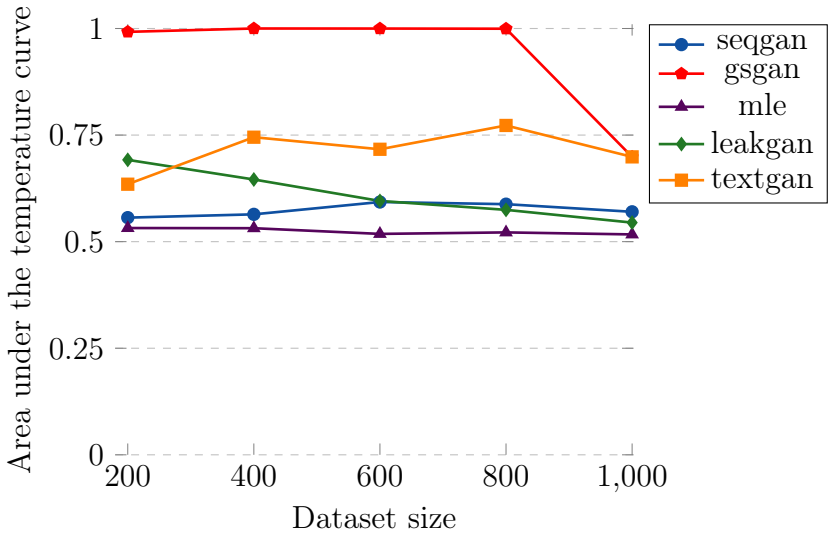


Figure 5.3: Experiment results on EMNLP NEWS dataset

Here, while LeakGAN seems to be producing more diverse data when the data size increases, SeqGAN seems to be performing quite acceptable compared to TextGAN and LeakGAN. Figure 5.3 sums up these two areas. It can be observed that GSGAN is approaching lower values when the size is 1000. Similar to figure 5.2, representation for TextGAN in figure 5.3 is misleading given its very low quality. So, the need for insights from area graphs related to individual scores is proven here as well. Apart from those facts, we can see the MLE based model model as the best performing model along with SeqGAN and LeakGAN.

5.2.5 Experiments with Geometry Questions

Doing the experiments above serves the purpose of identifying the appropriate models for text generation that will improve the performance as the dataset size grows. We conducted experiments to generate geometry questions so that the limitations we faced during our task of deep semantic parsing can be alleviated. However, in our experiments with this dataset, none of the models above could produce meaningful enough questions.

The dataset we have for the geometry domain consists of a huge variance while being very limited in size (226 sentences). Moreover, in contrast to above datasets, geometry questions need to be constrained. For instance, the entities

should have the same meaning across the sentence and their lengths (if any) should be meaningful. The experiment results showed no sign that this is learnt by the models. So, we claim that this dataset, at this stage, is not currently capable for being used as a training dataset for data augmentation with current neural models.

5.3 Discussion

In this work, we studied the applicability of state of the art text generation models on low resource domains. We examined the vanilla MLE language model [77], and four language GANs (SeqGAN [10], TextGAN [11], LeakGAN [12], and GSGAN [13]) using small datasets extracted from the datasets used in Texygen. We experimented with two datasets where one has shorted sentences than the other. When it comes to dataset sizes below 1000, we observed that GSGAN and TextGAN perform worse than other models on both datasets. Hence, based on our experiments, we can claim that GSGAN and TextGAN are not suitable to be applied on low resource domains. On the other hand, SeqGAN and LeakGAN showed fairly similar behavior in both datasets. Interestingly, the MLE based model turned out to be best performing model on both datasets despite its simplicity compared to the considered language GANs. This proves the claim made by Caccia et al. [14] that MLE based models are underestimated is true for low resource datasets as well. However, we saw that the final graphs with the sum of areas are not sufficient and can be misleading. Hence, we suggest filtering the models which perform acceptably referring the graphs indicating the area scores under (1-testBLEU) and selfBLEU axes, and then ranking those filtered models from the final graph.

We extended Texygen [77] (i) to be capable of evaluating text generation models more accurately using Temperature Sweep [14] along with the capability to analyze graphs related to individual scores, and (ii) to be capable of evaluating the trend of the performance of text generation models with dataset size increasing.

There exist a plethora of language GANs, we evaluated only 4 of those in

this study. To get a much broader and a generalized insight, setting up other language GANs in Texygen is necessary so that the experiments can be carried out in similar contexts. We only experimented for sizes under 1000, and we kept the size of the testing size (200) throughout the experiments. It would be interesting if we find a way to determine these values based on the qualities of the reference dataset. Intuitively, if a dataset is more complex (higher vocabulary, longer average length, lower selfBLEU, etc.) than another dataset, we need more data to start generating data for the first dataset than the second dataset. This was apparent from our experiments too in that EMNLP NEWS dataset needed more data than 1000 to become equal to the raw (unnormalized) area scores of the models which were trained with the COCO dataset. We believe it would be useful if we could come up with a mechanism to decide on a minimum size of data needed for a given reference set to generate more data. One approach to this would be fixating on a raw area score and increasing the dataset size until the model reaches that score. Experimenting this with a set of datasets with distinguishable features such as vocabulary size, average sequence length, SelfBLEU would make it possible to get some insight on the effect of these features on the quality and the diversity of the samples.

Chapter 6

CONCLUSION AND FUTURE WORK

Existing systems for semantic parsing of geometry problems use language-specific NLP tools and hard-coded parsing rules. Hence, they are not extensible to other languages. Also, maintainability of these systems is difficult as alterations of the parsing rules need human effort. Existing systems have resorted to this approach mainly due to lack of data in the geometry domain.

In this research, we implement a language-independent deep semantic parser for geometry problems. We did not use any NLP tool or any hard-coded parsing rules. Still our system shows a performance comparable to the performance of the existing systems. We model the task of semantic parsing as a question answering task. This way of modeling facilitates extraction of dynamic counts of rules per sentence. This also provisions the capability to extract facts even after the initial extraction, which is beneficial especially in geometry domain. For example, if this system is used to feed facts for an automatic solver, missing facts can even inhibit the solver from progressing. With our system, the solver gets the capability to extract facts it needs on-demand. We use memory networks for this task. We introduce structural changes to make them work in low resource domains. To facilitate multiple facts for a given query, we introduce a mechanism to control the focus areas of the model while retrieving facts.

Due to the challenges we faced in applying deep learning techniques on low resource domains, we also analysed data augmentation techniques for low resource domains. We analysed current state-of-the-art language generative models and the evaluation metrics used on them. Language Generative Adveserial Nets (Language GANs) are the claimed to be performing better than the traditional autoregressive approaches. However, based on the study by Caccia et al. [14], the evaluation used for the evaluations are not accurate. They showed that, despite the complexity of language GANs, still traditional approaches outperformed lan-

guage GANs. In our research, we show that the approach suggested by Caccia et al. [14] is not sufficient for accurate evaluations and we enhance that approach. Moreover, Language GANs are evaluated on large corpuses. We improved the existing platform for comparing these language models, Texygen [77], to include our enhanced metrics to evaluating language models and we ran the experiments with small-seed corpora. Our results were in the same line with Caccia that traditional autoregressive models performed better than the state-of-the-art language models. However, we suspect that the complexity of language GANs may also have an effect on the results. While autoregressive models are simple, language GANs have very complex structures. We suspect that their low permance in low resource domains might also be due to their inherent complexities. We leave the analysis of this issue to future work.

References

[1] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1476, 2015.

[4] Keith Jones. Issues in the teaching and learning of geometry. In *Aspects of teaching secondary mathematics*, pages 137–155. Routledge, 2003.

[5] Rahul Singhal, Martin Henz, and Kevin McGee. Automated generation of geometry questions for high school mathematics. In *CSEDU (2)*, pages 14–25, 2014.

[6] Chamupathi Mendis, Dhanushka Lahiru, Naduni Pamudika, Supun Madushanka, Surangika Ranathunga, and Gihan Dias. Automatic assessment of student answers for geometric theorem proving questions. In *2017 Moratuwa Engineering Research Conference (MERCon)*, pages 413–418. IEEE, 2017.

[7] Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. Synthesis of solutions for shaded area geometry problems. In *The Thirtieth International Flairs Conference*, 2017.

[8] Mrinmaya Sachan and Eric Xing. Learning to solve geometry problems from natural language demonstrations in textbooks. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 251–261, 2017.

[9] Risto Miikkulainen. Subsymbolic case-role analysis of sentences with embedded clauses. *Cognitive Science*, 20(1):47–73, 1996.

[10] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[11] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4006–4015. JMLR. org, 2017.

[12] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

[14] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv preprint arXiv:1811.02549*, 2018.

[15] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[16] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2011.

[17] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[22] Paul J Werbos et al. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[24] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[25] Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. Encoding source language with convolutional neural network for machine translation. *arXiv preprint arXiv:1503.01838*, 2015.

[26] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*, 2015.

[27] Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106*, 2015.

[28] Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1041–1044. ACM, 2016.

[29] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.

[30] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[31] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[34] Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*, pages 801–809, 2011.

[35] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[36] Samuel R Bowman, Christopher Potts, and Christopher D Manning. Recursive neural networks for learning logical semantics. *CoRR, abs/1406.1827*, 5, 2014.

[37] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, 2014.

[38] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[39] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[40] Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

[41] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[42] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387, 2016.

[43] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[44] Roee Aharoni and Yoav Goldberg. Morphological inflection generation with hard monotonic attention. *arXiv preprint arXiv:1611.01487*, 2016.

[45] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[46] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[47] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in neural information processing systems*, pages 899–907, 2013.

[48] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[49] P Kingma Diederik, Max Welling, et al. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[50] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[51] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using aï£ij laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[52] Radu Florian, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. Factorizing complex models: A case study in mention detection. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics, 2006.

[53] Radu Florian, John F Pitrelli, Salim Roukos, and Imed Zitouni. Improving mention detection robustness to noisy input. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 335–345. Association for Computational Linguistics, 2010.

[54] Jing Jiang and ChengXiang Zhai. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120, 2007.

[55] Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 419–426. Association for Computational Linguistics, 2005.

[56] Ralph Grishman. Information extraction: Techniques and challenges. In *International summer school on information extraction*, pages 10–27. Springer, 1997.

[57] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106, 2003.

[58] Yee Seng Chan and Dan Roth. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the*

Association for Computational Linguistics: Human Language Technologies-Volume 1, pages 551–560. Association for Computational Linguistics, 2011.

[59] Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, 2014.

[60] Makoto Miwa and Yutaka Sasaki. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, 2014.

[61] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*, 2016.

[62] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[63] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[64] Meishan Zhang, Yue Zhang, and Guohong Fu. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740, 2017.

[65] Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. A neural probabilistic structured-prediction model for transition-based dependency parsing. 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)*, pages 1213–1222, 2015.

[66] Taro Watanabe and Eiichiro Sumita. Transition-based neural constituent parsing. 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), pages 1169–1179, 2015.

[67] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[68] Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*, 2017.

[69] Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, 2014.

[70] Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial text generation via feature-mover's distance. In *Advances in Neural Information Processing Systems*, pages 4671–4682, 2018.

[71] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[72] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[73] Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

[74] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165, 2017.

[75] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[76] Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. Neural text generation: past, present and beyond. *arXiv preprint arXiv:1803.07133*, 2018.

[77] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. *arXiv preprint arXiv:1802.01886*, 2018.

[78] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. Connectionist models and their implications: Readings from cognitive science. chapter a learning algorithm for boltzmann machines, 1988.

[79] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*, 2015.

[80] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*, 2015.

[81] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.

[82] Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*, 2015.

[83] Yi Luan, Yangfeng Ji, and Mari Ostendorf. Lstm based conversation models. *arXiv preprint arXiv:1603.09457*, 2016.

[84] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

[85] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.