Understanding and Generating Multi-Sentence Texts

Rik Koncel-Kedziorski

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Hannaneh Hajishirzi, Chair

Gina-Anne Levow, Chair

Emily M. Bender

Program Authorized to Offer Degree:

Linguistics

University of Washington

Abstract

Understanding and Generating Multi-Sentence Texts

Rik Koncel-Kedziorski

Chairs of the Supervisory Committee:
Professor Hannaneh Hajishirzi
Computer Science and Engineering
Professor Gina-Anne Levow
Linguistics

English is often found in units comprised of multiple sentences, but synthesizing information across sentence boundaries, whether for understanding or generation, is a difficult challenge for natural language processing algorithms. Techniques for such synthesis, however, are necessary for improved language understanding and have the potential to transform downstream applications including dialog systems, question answering, and educational technologies. In this thesis, I investigate techniques for understanding and generating multi-sentence natural language texts.

As a first step toward general cross-sentence reasoning, I will describe a model for solving open-world math word problems. The model treats word problem texts as semantically-enhanced equation trees using a recursive semantic structure of quantities and generates possible solutions with an integer linear programming approach. Local and global information from the text is combined, and the model learns from data how to select the maximum scoring tree to answer each problem.

Continuing in the math word problem domain, I present an editing method for automatically customizing math word problems to meet thematic constraints. This technique preserves the complex document structure of human authored text, editing in a globally coherent and syntactically informed way. Additionally, this model improves on previous thematic generation approaches by automatically building an understanding of theme from an arbitrary text. Reusing the existing syntactic and semantic relationships of the human authored text to preserve its mathematical meaning, my method can produce novel and coherent thematic word problems in English.

Finally, I outline a model for generating multi-sentence texts from knowledge graphs using an innovative neural encoding, and provide evidence that knowledge graphs can help structure the generation of longer English texts. My novel graph transforming encoder extends the recent transformer model for text encoding to graph-structured inputs. The overall model learns to encode the input graph and output text in an end to end fashion. Human and automatic evaluation show that relational knowledge improves generated text.

Acknowledgement

I want to express my unending gratitude and thanks to my research advisor Hannaneh Hajishirzi, without whom I would very likely have failed at this endeavor. Hanna took a chance on me when I had nothing to offer but an arrogance borne of ignorance, and through her steadfast and patient guidance taught me the skills of a trained research scientist. Thank you for always pushing me to achieve more than I could.

Many thanks also to my academic advisor, Gina-Anne Levow, whose sage and level-headed advice helped me navigate the complex waters of an interdisciplinary education, and to Emily M. Bender, whose attention to detail helped keep me honest throughout my education.

Thanks as well to the many brilliant co-authors who contributed to my understanding of NLP and the process of producing high quality research: Ali Farhadi, Ashish Sabharwal, Oren Etzioni, Siena Ang, Subhro Roy, Aida Amini, Nate Kushman, Aaron Jaech, Mari Ostendorf, Kirby Conrod, Rachael Tatman, Ioannis Konstas, Luke Zettlemoyer, Ben Robaidek, Sachin Mehta, Mohammad Rastegari, Dhanush Bekal, Yi Luan, and Mirella Lapata.

I wish to acknowledge my mother, Kathleen Koncel, to whom I owe my grit and finesse, and my father, Henry Kedziorski, to whom I owe my love of math and puns. I should acknowledge my brother Michael as well, why not. Thank you brother Michael. You're a fine brother.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

'Understanding' is a difficult term to define. What does it mean to understand something? Rather, what is true of the person who understands? Perhaps it is better to ask, "How does one demonstrate their understanding?" Understanding can be demonstrated in myriad ways, but consider these three: a person who understands a concept or situation can *describe* their understanding; they can *answer questions* about the content of their understanding; and finally, once they have mastery of their subject, a person can *synthesize* novel information from different sources of understanding.

Based on these assumptions about understanding, this thesis considers whether a computer can understand multi-sentence texts. Specifically, I offer models for text understanding and generation that answer questions about about a text, synthesize a novel text from diverse input sources, and describe the information contained in a knowledge graph.

In recent years, advances in hardware and modeling power, coupled with the construction of large datasets for common tasks, have resulted in significant improvements for natural language understanding and generation. Tasks such as question answering [Rajpurkar et al., 2016] , image captioning [Feng and Lapata, 2010], and dialog response generation [Li et al., 2015] are approaching human levels of accuracy and fluency for English language inputs. These challenging tasks are limited in their scope, however, as they primarily deal with sentences in isolation. In the case of

image captioning and dialog response generation, the target response is a single sentence. In the case of question answering, there are few datasets which require synthesizing information across sentence boundaries, and thus models can succeed without such capacity.

Yet understanding and even generating multi-sentence texts is of great interest from both applied and theoretical positions. Theoretically, multi-sentence texts force us to look beyond the formal *standing or semantic meaning* of a sentence and to confront it's contextualized *speaker or pragmatic meaning* [Quine, 1969; Grice, 1968]. The variety of meaning-influencing contexts in which a sentence can occur is quite large, and the goals or intended meaning of short texts vary. In this work, I focus on 2 specific domains (math word problems and scientific abstracts) in an effort to gain insights into general techniques for understanding or generating multi-sentence English language texts.

From an applied standpoint, a better technique for processing multi-sentence texts could yield significant improvements to current systems where cross-sentence contextualization is important, such as machine translation and information extraction. Applications of multi-sentence text generation include educational technologies (e.g. customizing educational materials to the thematic preferences of a student) and entertainment (for example, generating on-the-fly conversations in a video game world based on the current game state). The methods provided in this thesis can assist in these applications and more.

The goal of this work is to help develop techniques for understanding and generating multi-sentence texts. Like much research on pragmatic understanding, I focus on constrained domains: in this instance, math word problems and scientific texts. In Chapter 3, I describe a model for automatically solving open-world math word problems using cross-sentence reasoning. This model has immediate practical applications to automatic tutoring systems, while also providing insight into general techniques for multi-sentence question answering. Building on this understanding, in Chapter 4, I describe a method for automatically customizing math word problems to meet thematic constraints using editing techniques. Smart editing allows for leveraging the complex document

structure of human authored text, improving the coherence of the system's outputs. Finally, in Chapter 5, I provide a method for generating multi-sentence English texts from knowledge graphs using innovative neural encoding techniques. Knowledge graphs are ubiquitous in computing and I show that they have the capacity to serve as *document plans* for longer texts.

Two primary research questions motivate this thesis. First, what kind of formal symbolic or semi-formal structures can be used to partially express the pragmatic meaning of a text? In Chapter 3, I outline *semantically-enhanced equation trees*, which leverage a recursive semantic structure that allows for building equation trees from text. In Chapter 4, I propose that syntactic and mathematical relationships in an existing text can serve as a pragmatic scaffolding, permitting semantic edits without impacting pragmatic meaning. In Chapter 5, I show that existing knowledge graphs can be used as document plans for generating short texts. These various structures are shown to be helpful in the tasks where they are used.

The second focus of this work deals with mixing local and global information. Is it useful when handling multi-sentence texts to combine local information, such as syntactic or adjacency information, with global information, such as thematic similarity across several sentences or long-distance relationships in a text? In Chapter 3, I combine local and global classifiers to rank trees, and provide ablations which show the superiority of this mixing method compared to local or global information alone. In Chapter 4, I show that maximizing the likelihood of local syntactic relationships and global semantic relationships improves text editing outcomes. In Chapter 5, a global graph encoding is combined with a local language model via an attention mechanism to determine the best generated text sequence. The experiments outlined in this thesis provide evidence that we should incorporate both local and global information in our modeling going forward.

1.1 Overview of Previous work

Previous work on multi-sentence text understanding has focused on the task of word problem solving due to its simplicity and the easily verifiable nature of a model's solution. However, many previous works focus on number word problems, geometry problems, or other closed-world domains where a limited and math-specific set of nouns and verbs express mathematical relations [Shi et al., 2015; Mitra and Baral, 2015; Roy and Roth, 2015; Zhou et al., 2015; Seo et al., 2015]. This thesis addresses open-domain word problems with arbitrary subjects and events. A few previous works address open-domain word problems. Hosseini et al. [2014] provide a method for solving addition or subtraction problems using a semantics of *entities* corresponding to individual units. These entities can occupy different *containers*, corresponding to sets. They offer a technique of verb-categorization to learn how the verbs in a word problem text change the number of entities that belong to each container. The current work improves upon their model by using an enriched semantic representation that includes quantifiers, adjectives, and adverbs when updating the state of containers. Additionally, I am able to cover a larger set of mathematical operations by modeling equations with *semantically-enhanced equation trees*.

Another high-impact previous work is that of Kushman et al. [2014], who use template-induction and alignment techniques to solve open-domain word problems. However, I show this method to be brittle and unable to generalize to equations not seen during training. My method combines bottom up and top down constraints to search a large space of possible solutions, including structures not among the training instances.

Previous work in text editing has focused solely on the single-sentence task. Often such work aims to meet constraints on simplicity or seeks to introduce paraphrases. To my knowledge, the current work is the first to offer an editing technique for multi-sentence texts. Additionally, the thematic constraints used here are parameterized by an arbitrary topic, and are thus more general than previous constraints on sentence complexity. Because the proposed editing techniques are

applied to generating thematic math word problems, I compare immediately to the work of Polozov et al. [2015], who develop 3 themes in which math word problems can automatically be written. These authors note that the development of these themes took 1-2 months each. By contrast, my model develops a notion of theme automatically from a text such as a movie script, allowing for math problem generation in arbitrary themes with minimal specialist involvement.

The model I offer for generating multi-sentence texts from knowledge graphs is, to my knowledge, the proposed is the first model of its kind. This work can be considered a special case of concept-to-text generation, which has been well studied [Barzilay and Lapata, 2005; Liang et al., 2009; Kim and Mooney, 2010; Konstas and Lapata, 2013]. These earlier works often focused on smaller record generation datasets, but Mei et al. [2016] showed these datasets to be easy for modern neural models. More recently, Lebret et al. [2016] generate the first sentence of a Wikipedia entry from the associated infobox, and Wiseman et al. [2017] study the difficulty of applying neural models to a similar data-to-text task. The current work looks at the specific case of generating from knowledge graphs. Knowledge graphs are a very general format which can be extracted cheaply from a variety of texts, images, databases, or even manually constructed. In this work, scientific knowledge is encoded with a novel Graph Transformer architecture and used to generate document abstracts. I show that relational knowledge can indeed be leveraged for text generation, and that the proposed encoder outperforms other recent architectures for knowledge encoding on this task.

1.2 Overview of solutions

The solutions I document in this thesis follow a similar thread for both text understanding and generation. The overarching argument is as follows: to best model the pragmatic information necessary for understanding and generating multi-sentence texts we must develop algorithms that combine local information about each relationship of interest (whether intra- or inter-sentential) with global structural information. Local information may refer to sentence- or phrase-level information,

or it may refer to information unique to a particular node in the pragmatic structure (i.e. equation tree or document plan). Global information is likewise defined as document-level information, or discriminating characteristics of the complete pragmatic structure under consideration.

1.2.1 Understanding

For the task of solving multi-sentence open-domain algebra word problems, I offer the ALGES model, which increases the scope of solvable problems compared to Hosseini et al. [2014] and drastically improves robustness compared to Kushman et al. [2014]. Each math word problem is modeled as semantically-enhanced equation tree, where each node of the tree corresponds to a recursive structure called a *qset*. *Qsets* record semantic information about the quantities involved in a math word problem. They are derived from syntactic, coreference, and domain-specific textual information. Importantly, the mathematical combination of two qsets yields another qset, so that the model can treat atomic and composed quantities similarly. With this formalism, the model builds a collection of possible semantically-enhanced equation trees from a parse of the word problem text. The space of trees is defined using an integer linear programming approach. ALGES learns to score local combinations of qsets, producing *local scores*. These are combined with a *global score* for each equation tree which captures its likelihood based on learned features and violated ILP constraints. The maximum scoring tree is evaluated to provide an answer for the problem. I show that this method outperforms previous approaches, and that the simple biases introduced by the model significantly improve generalization.

1.2.2 Generation

I explore two avenues for automatically generating multi-sentence texts. First, I provide a general technique for editing human-authored texts directly so as to produce new texts with desirable thematic properties. I demonstrate this technique by customizing math word problems to fit an

arbitrary theme of student interest. Given an algebra word problem and an arbitrary thematic text (such as a movie script), the proposed ThemeRewriter model identifies the most thematic words and characters from this document. It then constructs a collection of candidate rewrites of the word problem text by substituting thematic elements for those components of the original text which do not affect the pragmatic (i.e. mathematical) meaning of the document. Informed by my previous work in text understanding, I presume that syntactic and intra- and cross-sentential semantic relationships are the strongest contributors to document coherence and the pragmatic (in this case, mathematical) meaning of the text. Under this assumption, ThemeRewriter aims to balance its thematic substitutions against the text's existing syntactic and semantic relationships. Because the space of possible rewrites is intractable, a beam-like search procedure is used to effectively search the space. At each timestep, a single word is changed to the most thematic word that fits in the syntactic and pragmatic structure of the existing text. By iterating this procedure, ThemeRewriter produces the rewrite which is most aligned with the provided thematic text while maintaining the mathematical intention of the original word problem.

As a second approach to multi-sentence generation, I propose a neural technique for generating multi-sentence texts from graphical knowledge representations, called GraphWriter. Graph-structured knowledge is ubiquitous in computing. These inputs can be extracted and compiled from a variety of text and non-text sources, making it an ideal medium for representing a *document plan* or representation of the pragmatic meaning of a text.

At the core of GraphWriter is a *graph transforming* encoder which extends the recent transformer model for text encoding to graph-structured inputs. The encoded graph representation acts as input to an neural encoder-decoder style model for sequence generation. Using dataset of 40k knowledge graphs extracted from scientific abstracts, GraphWriter learns to encode a provided knowledge graph and decode the abstract text, copying text from input nodes as needed. Beam search allows for approximate inference of the most probable output sequence. Detailed human evaluation shows that GraphWriter is capable of using the relational knowledge of the input to improve generated text

compared to models which cannot leverage this knowledge. I also show that the graph transforming encoder outperforms other recent graph encoding techniques on this task.

Chapter 2

Related Work

This chapter discusses relevant background and related work.

2.1 Natural Language Understanding

A great deal of work has been done in natural language understanding, but work that focuses on reasoning across sentence boundaries is much more limited. Some important directions of this work are coreference resolution (for an overview, see Lee et al. [2017]) and multi-hop question answering (for an overview, see Welbl et al. [2018]).

The current work is related to (semantic) parsing, which aims to map natural language sentences to formal meaning representations. One direction of this research involves the expert construction of precision grammars for specific languages [Pollard and Sag, 1994; Flickinger, 2000, 2011; Bender et al., 2015]. Another aims to learn to map text to logical forms from annotated data alone [Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Ge and Mooney, 2006; Kwiatkowski et al., 2010]. The current work combines elements of both of these lines of research.

More closely related is work on language grounding, whose goal is the interpretation of a sentence in the context of a world representation [Branavan et al., 2009; Liang et al., 2009; Chen et al., 2010; Bordes et al., 2010; Hajishirzi et al., 2011; Matuszek et al., 2012; Hajishirzi et al., 2012; Artzi and Zettlemoyer, 2013; Koncel-Kedziorski et al., 2014; Seo et al., 2014; Hixon et al., 2015]. Much work on language grounding considers the problem of generating natural language descriptions of images [Farhadi et al., 2010; Kulkarni et al., 2011; Yatskar et al., 2014; Feng and Lapata, 2010]. A more related strain to the current work focuses on uncovering *speaker*- rather than *sentence*- meaning [Bender et al., 2015]. Efforts toward this goal include grounding sports commentaries in event logs [Chen and Mooney, 2008; Chen et al., 2010; Bordes et al., 2010; Hajishirzi et al., 2011; Liang et al., 2009; Koncel-Kedziorski et al., 2014], natural language instructions in executable robot control commands [Branavan et al., 2009; Vogel and Jurafsky, 2010; Matuszek et al., 2012; Artzi and Zettlemoyer, 2013], and geometry word problems and diagrams [Seo et al., 2014].

However, while most previous work considered individual sentences in isolation, solving word problems often requires reasoning across the multi-sentence discourse of the problem text. Recent efforts in the math domain have studied number word problems [Shi et al., 2015], logic puzzle problems [Mitra and Baral, 2015], arithmetic word problems [Hosseini et al., 2014; Roy and Roth, 2015], algebra word problems [Kushman et al., 2014; Zhou et al., 2015], and geometry word problems [Seo et al., 2015]. I discuss in more detail below two pioneering works closely related to the current work.

Hosseini et al. [2014] solve elementary addition and subtraction problems by learning verb categories. They ground the problem text to a semantics of *entities*, such as cats or coins, and *containers* which hold entities, such as cars or coat pockets. Their method decides if quantities are increasing or decreasing in a container based upon the learned verb categories. While relying only on verb categories works well for $+$ and $-$, modeling $*$ or $/$ requires going beyond verbs. For instance, "Tina has 2 cats. John has 3 more cats than Tina. How many cats do they have together?"

and “Tina has 2 cats. John has 3 times as many cats as Tina. How many cats do they have together?” have identical verbs, but the indicated operation (+ and * resp.) is different. ALGES makes use of a richer semantic representation which facilitates deeper learning and a wider scope of application, solving problems involving the $+, -, /,$ and $*$ operators (see Table 6).

Kushman et al. [2014] introduce a general method for solving algebra problems. This method can align a word problem to a system of equations with one or two unknowns. They learn a mapping from word problems to equation templates using global and local features from the problem text. However, the large space of equation templates makes it challenging for this model to learn to find the best equation directly, as a sufficiently similar template may not have been observed during training. Instead, my method maps word problems to equation trees, taking advantage of a richer representation of quantified nouns and their properties, as well as the recursive nature of equation trees. These allow ALGES to use a bottom-up approach to learn the correspondence between spans of texts and arithmetic operators (corresponding to intermediate nodes in the tree). ALGES then scores equations using global structure of the problem to produce the final result.

The current work is also related to research in using ILP to enforce global constraints in NLP applications [Roth and Yih, 2004]. Most previous work [Srikumar and Roth, 2011; Goldwasser and Roth, 2011; Berant et al., 2014; Liu et al., 2015] utilizes ILP as an inference procedure to find the best global prediction over initially trained local classifiers. Similarly, ALGES uses ILP to enforce global and domain specific constraints. However, ALGES uses ILP to form candidate equations which are then used to generate training data for classifiers. The current work is also related to parser re-ranking [Collins, 2005; Ge and Mooney, 2005], where a re-ranker model attempts to improve the output of an existing probabilistic parser. Similarly, the global equation model designed in ALGES attempts to re-rank equations based on global problem structure. The base parser produces a set of candidate parses for each input sentence, with associated probabilities that define an initial ranking of these parses. A second model then attempts to improve upon this initial ranking, using additional features of the tree as evidence.

2.2 Natural Language Generation

The current work explores two tactics for generating longer natural language texts. On the one hand, I develop a method for thematic rewriting of existing texts subject to constraints. This follows a broader trend of editing in natural language generation that I discuss in Section 2.2.1. On the other, I explore the potential for using graphical knowledge representations in long text generation. I discuss the relationship of the current work to related trends in Section 2.2.2.

2.2.1 Editing

My approach to generating narrative math word problems by editing is related to the previous work on math word problem generation described in Polozov et al. [2015], story generation (e.g., McIntyre and Lapata [2010]) and sentence rewriting (e.g., text simplification [Xu et al., 2016]). I will cover each of these comparisons below in detail. To my knowledge, I am the first to introduce a rewriting formulation for generating multi-sentence texts.

Polozov et al. [2015] automatically generate math word problems tailored to a student's interest using Answer Set Programming to satisfy a collection of pedagogical and narrative requirements. Starting from the equation they formulate a content plan in the form of a logical graph using events and entities from an ontology. Semantic coherence based on predefined *discourse tropes* is applied as an extra set of constraints in the graph; the resulting text is generated using hand-crafted templates.

This method naturally produces highly coherent, personalized story problems that meet pedagogical requirements, at the expense of building the thematic ontologies and discourse constraints by hand. According to Polozov et al. [2015] building small thematic ontologies of types, relations, and discourse tropes (100-200 entries) for each of only 3 literary settings took 1-2 person months. In contrast, my model acquires the theme automatically from an input unprocessed document collection. Moreover, my method can operate on arbitrary input documents and seed thematic

document collections. This is in contrast with the previous work which is limited by both the size of the semantic ontologies and the document templates available.

Because I focus on narrative math word problems, in which a story is used to describe a mathematical equation, my work is related to previous works on story generation. Recent methods in story generation first synthesize candidate plots for a story and then compile those plots into text. Li et al. [2013] use crowdsourcing to build plot graphs. McIntyre and Lapata [McIntyre and Lapata, 2009, 2010] address story generation through the automatic deduction and reassembly of scripts [Schank and Abelson, 1977], or structured representations of events and their participants, and causal relationships involved. Leveraging the automatic script learning methods of Chambers and Jurafsky [2009], McIntyre and Lapata [2010] learn candidate entity-centered plot graphs, or possible events involving the entity and an ordering between these events. They use a genetic algorithm to mix and mutate plot graphs to produce stronger plot candidates. Then plots are compiled into stories through the use of a rule-based text surface realizer [Lavoie and Rambow, 1997] and reranked using a language model.

While contributing to variety in the production of new plots, event co-occurrence as a content planner component oversimplifies the definition of plot, reducing to less coherent or more mundane stories compared to human-authored. Worse, genetic algorithms can make changes which reduce the semantic soundness of the plot, a factor crucial for the domain of math word problems and their potential solvability. Responding to this outcome, the current work aims to preserve much of the original human-authored plot and only deviates from it subject to semantic, syntactic, and discourse coherence constraints.

Finally, there is related work in text simplification [Wubben et al., 2012; Kauchak, 2013; Zhu et al., 2010; Vanderwende et al., 2007; Woodsend and Lapata, 2011b], sentence compression [Filippova and Strube, 2008; Rush et al., 2015], and paraphrasing [Ganitkevitch et al., 2013; Chen and Dolan, 2011; Ganitkevitch et al., 2011]. All these tasks are focused on rewriting sentences under a predefined set of constraints, such as simplicity. Different rule-based and data-driven approaches are

introduced by Petersen and Ostendorf [2007], Vickrey and Koller [2008], and Siddharthan [2004]. Most data-driven approaches take advantage of machine translation techniques, use source-target sentence pairs, and learn rewrite operations [Yatskar et al., 2010; Woodsend and Lapata, 2011a], or use additional external paraphrasing resources [Xu et al., 2016]. These approaches primarily focus on editing a single sentence in isolation, rather than a multi-sentence text. Additionally, their aim is to produce semantically equivalent texts, whereas theme-transfer aims to modify the semantics while maintaining the intended purpose of the text (here, the mathematical equation). Lastly, other editing techniques rely on source-target pairs for learning, which do not exist for arbitrary themes.

2.2.2 Text Generation from Graphs

In this section, I describe the most closely related works to my own dealing with text generation from data and graph structures. My work falls under the larger scope of concept-to-text generation. Earlier works in this area often focused on smaller record generation datasets such as WeatherGov and RoboCup [Barzilay and Lapata, 2005; Liang et al., 2009; Kim and Mooney, 2010; Konstas and Lapata, 2013]. Recently, however, Mei et al. [2016] showed how neural models can achieve strong results on these standards, prompting researchers to investigate more challenging domains. I offer a new more challenging concept-to-text dataset, the AGENDA dataset described in Section 5.1.

Lebret et al. [2016] tackles the task of generating the first sentence of a Wikipedia entry from the associated infobox, which contains relevant entities and relations. They provide a large dataset of such entries and a language model conditioned on tables. My work and data extend this to the multi-sentence setup. In AGENDA, relations can extend across sentence boundaries.

Wiseman et al. [2017] offer a study of the difficulty of applying neural models to the data-to-text task. They introduce a large dataset where a summary of a basketball game is paired with two tables of relevant statistics. To generate game summaries from the tables of statistics, a model must learn content selection, content planning, and surface realization. Wiseman et al. [2017] show that neural models struggle to compete with template based methods at this task. My work focuses on the

content planning and surface realization steps of this process. The automatically extracted nature of knowledge graphs used here requires that, rather than content selection, a model learns to tolerate noise.

I show that modeling knowledge as a graph improves generation results, connecting the current work to other graph-to-text tasks such as generating from Abstract Meaning Representation (AMR) graphs. Konstas et al. [2017] provide the first neural model for this task, and show that pretraining on a large dataset of noisy automatic parses can improve results. However, they do not directly model the graph structure, relying on linearization and sequence encoding instead. Current works improve this through more sophisticated graph encoding techniques. Marcheggiani and Perez-Beltrachini [2018] encode input graphs directly using a graph convolution encoder [Kipf and Welling, 2017]. My model extends the graph attention networks of Veličković et al. [2018], a direct descendant of the convolutional approach which offers more modeling power and has been shown to improve performance. Song et al. [2018] uses a graph LSTM model to effect information propagation. At each timestep, a vertex is represented by a gated combination of the vertices to which it is connected and the labeled edges connecting them. Beck et al. [2018] use a similar gated graph neural network. Both of these gated models make heavy use of label information, which is much sparser in scientific knowledge graphs than in AMR. Moreover, AMR graphs are denser, rooted, and connected, whereas the knowledge graphs studied here lack these characteristics. For this reason, I focus on attention-based models such as Veličković et al. [2018], which impose fewer constraints on their input. They also make use of Levi graph transformations [Levi, 1942] which convert labeled graphs to bipartite unlabeled graphs without loss of information. I follow many of the same graph conversions used in Beck et al. [2018] when preparing my own model's input.

Finally, my work is related to Wang et al. [2018] who offer a method for generating scientific abstracts from titles. Their model uses a gated rewriter network to write and revise several drafts outputs in several sequence-to-sequence steps. While I operate in the same general domain as this work, my task setup is ultimately different due to the use of extracted information as input. I argue

that my task setup improves the task defined in Wang et al. [2018] due to the stronger connection to previous work, and the fact that the use cases for a knowledge-to-text model are more immediately identifiable than those of a title-to-document model.

Chapter 3

Reasoning Across Sentences

English-language grade-school algebra word problems read like short narratives (see Figure 3.1).[1] It first describes a partial world state consisting of characters, entities, and quantities. As the description progresses, it elaborates on this world state by updating the condition of a referent or explicating relationships between referents. The conclusion then poses a simple question about the world described.

A 4th grader may struggle with the mechanics of the mathematics required to solve a word problem, but has little trouble understanding what is going on in the story utilizing extensive world knowledge, large vocabulary, word-sense disambiguation, coreference resolution, mastery of syntax, and the ability to combine individual sentences into a coherent mental model. For an algorithm, the problem is the opposite: the challenge for an NLP system is to "make sense" of the narrative, which may refer to arbitrary activities like renting bikes, collecting coins, or eating cookies.

Previous work coped with the open-domain aspect of algebraic word problems by learning how subsequent verbs in the text affect quantities [Hosseini et al., 2014] or by learning templates that cover equations of particular forms [Kushman et al., 2014]. My experiments described in Section 3.6 that both approaches are brittle, particularly as training data is scarce in this domain,

[1]The work presented in this chapter was done with Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. It appears as Koncel-Kedziorski et al. [2015]

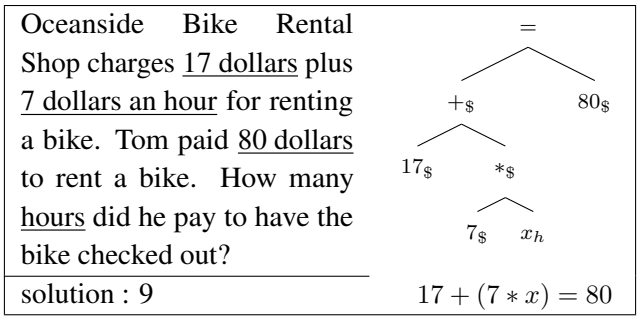

Figure 3.1: Example problem and solution

and the space of equations grows exponentially with the number of quantities mentioned in the math problem.

I introduce ALGES,[2] which maps an unseen multi-sentence algebraic word problem into a set of possible equation trees. Figure 3.1 shows an equation tree alongside the word problem it represents. ALGES generates the space of trees via Integer Linear Programming (ILP) [Schrijver, 1998], which allows it to constrain the space of trees to represent type-consistent algebraic equations satisfying as many desirable properties as possible. ALGES learns to map spans of text to arithmetic operators, to combine them given the global context of the problem, and to choose the "best" tree corresponding to the problem. The training set for ALGES consists of unannotated algebraic word problems and their solution. Solving the equation represented by such a tree is trivial. ALGES is described in detail in Section 3.2.

ALGES is able to solve word problems with single-variable equations like the ones in Figure 3.1. In contrast to Hosseini et al. [2014], ALGES covers $+, -, *$, and $/$. The work of Kushman et al. [2014] has broader scope but I show that it relies heavily on overlap between training and test data. When that overlap is reduced, ALGES is 15% to 50% more accurate than this system.

My contributions are as follows: (1) I formalize the problem of solving multi-sentence algebraic word problems as that of generating and ranking equation trees; (2) I show how to score the likelihood of equation trees by learning discriminative models trained from a small number of word

[2]The code and data is publicly available at `https://gitlab.cs.washington.edu/ALGES/TACL2015`.

problems and their solutions – without any manual annotation; and (3) I demonstrate empirically that ALGES has broader scope than the system of Hosseini et al. [2014], and overcomes the brittleness of the method of Kushman et al. [2014].

3.1 Setup and Problem Definition

Given numeric quantities V and an unknown x whose value is the answer sought, an *equation* over V and x is any valid mathematical expression formed by combining elements of $V \cup \{x\}$ using binary operators from $\mathcal{O} = \{+, -, *, /, =\}$ such that x appears exactly once. When each element of V appears at most once in the equation, it may naturally be represented as an *equation tree* where each operator is a node with edges to its two operands.[3] T denotes the set of all equation trees over V and x.

Problem Formulation. I address the problem of solving grade-school algebra word problems that map to single equations. A word problem w can be solved by selecting an equation tree t representing the correct mathematical computation for solving w from the set of all possible equation trees. Figure 3.1 shows an example of w with quantities underlined, and the corresponding tree t. Formally, I use a joint probability distribution $p(t, w)$ that defines how "well" an equation tree $t \in T$ captures the mathematical computation expressed in w. Given a word problem w as input, my goal is to compute $\tilde{t} = \arg\max_{t \in T} p(t|w)$.

An exhaustive enumeration over T quickly becomes impractical as problem complexity increases and $n = |V \cup \{x\}|$ grows. Specifically, $|T| > h(n) = n!\,(n-1)!\,(n-1)\,2^{n-4}$, $h(4) = 432, h(6) > 1.7M, h(8) > 22B$, etc. This vast search space makes it challenging for a discriminative model to learn to find $\tilde{t}$ directly, as a sufficiently similar tree may not have been observed during training. Instead, this method first generates syntactically valid equation trees, and then uses a bottom-up approach to score equations with a *local* model trained to map spans of text to math operators, and

[3]Problems involving simultaneous equations require combining multiple equation trees, one per equation.

a *global* model trained for coherence of the entire equation w.r.t. global problem text structure.

3.2 Overview of the Approach

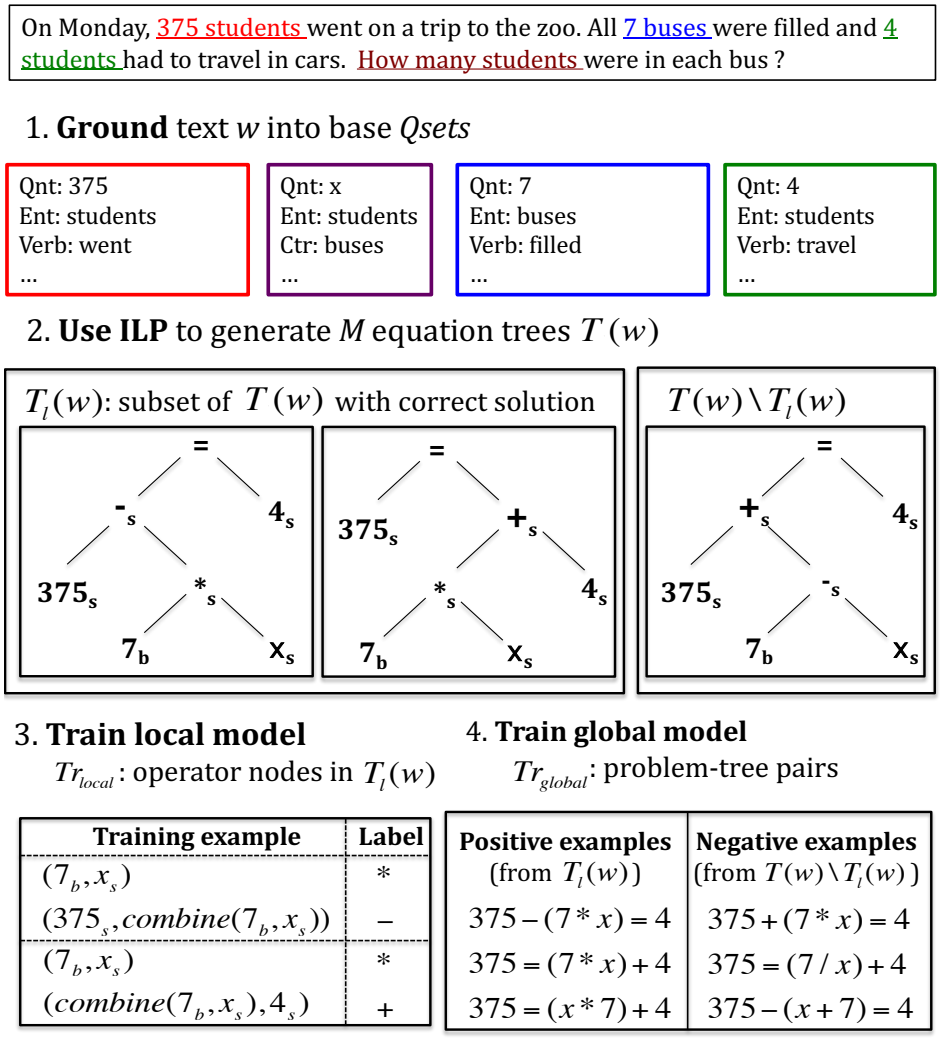

Figure 3.2: An overview of the process of learning for a word problem and its Qsets.

Figure 3.2 gives an overview of my method, also detailed in Figure 3.3. In order to build equation trees, I use a compact representation for each node called a *Quantified Set* or Qset to model natural language text quantities and their properties (e.g., '375 students' in '7 buses'). Qsets are

Learning (word problems W, corresponding solutions L):

1. For every problem-solution pair (w_i, ℓ_i) with $w_i \in W, \ell_i \in L$
   (a) $S \leftarrow$ Base Qsets obtained by Grounding text w_i and Reordering the resulting Qsets (Section 3.3)
   (b) $T_i \leftarrow$ Top M type-consistent equation tree candidates generated by ILP(w_i) (Section 3.4)
   (c) $T_{\ell_i} \leftarrow$ Subset of T_i that yields the correct numerical solution ℓ_i
   (d) Add to Tr_{local} features $\langle s_1, s_2 \rangle$ with label *op* for each operator *op* combining Qsets s_1, s_2 in trees in T_{ℓ_i}
   (e) Add to Tr_{global} features $\langle w, t \rangle$ labeled positive for each $t \in T_{\ell_i}$ and labeled negative for each $t \in T \setminus T_{\ell_i}$
2. $\mathcal{L}_{local} \leftarrow$ Train a local Qset relationship model on Tr_{local} (Section 3.5.1)
3. $\mathcal{G}_{global} \leftarrow$ Train a global equation model on Tr_{global} (Section 3.5.2)
4. Output local and global models $(\mathcal{L}_{local}, \mathcal{G}_{global})$

Inference (word problem w, local set relation model $\mathcal{L}_{local}$, global equation model $\mathcal{G}_{global}$):

1. $S \leftarrow$ Base Qsets obtained by Grounding text w_i and Reordering the resulting Qsets (Section 3.3)
2. $T \leftarrow$ Top M type-consistent equation tree candidates generated by ILP(w) (Section 3.4)
3. $t^* \leftarrow \arg\max_{t_i \in T} \left(\prod_{t_j \in t} \mathcal{L}_{local}(t_j|w) \right) \times \mathcal{G}_{global}(t|w)$, scoring each tree $t_i \in T$ based on Equation 3.1
4. $\ell \leftarrow$ Numeric solution to w obtained by solving equation tree t^* for the unknown
5. Output (t^*, ℓ)

Figure 3.3: Overview of this method for solving algebraic word problems.

used for tracking and combining quantities when learning the correspondence between equation trees and text.

Definition 1. Given a math word problem w, let $\mathbb{S}$ be the set of all possible spans of text in w, ϕ denote the empty span, and $S^\phi = S \cup \{\phi\}$. A **Qset** for w is either a base Qset or a compound Qset. A *base Qset* is a tuple $(ent, qnt, adj, loc, vrb, syn, ctr)$ with:

- $ent \in \mathbb{S}$: *entity* or quantity noun (e.g., 'student');

- $qnt \in \mathbb{R} \cup \{x\}$: ***number*** or quantity (e.g., 4 or x);
- $adj \subseteq \mathbb{S}^{\phi}$: ***adjectives*** for ent in w;
- $loc \in \mathbb{S}^{\phi}$: ***location*** of ent (e.g., 'in the drawer');
- $vrb \in \mathbb{S}^{\phi}$: ***governing verb*** for ent (e.g., 'fill');
- syn: ***syntactic*** and positional information for ent (e.g., 'buses' is in subject position);
- $ctr \subseteq \mathbb{S}^{\phi}$: ***containers*** of ent (e.g., 'Bus' is a container for the 'students' Qset).

Properties being ϕ indicates these optional properties are unspecified. A ***compound Qset*** is formed by combining two Qsets with a non-equality binary operator as discussed in section 3.3.

When a pair of Qsets are combined by the equality operator, they yield a ***semantically augmented equation tree***.[4] The example in Figure 3.2 has four base Qsets extracted from problem text. Each possible equation tree corresponds to a different recursive combination of these four Qsets.

Given w, ALGES first extracts a list of n base Qsets $S = \{s_1, \ldots, s_n\}$ (Section 3.3). It then uses an ILP-based optimization method to combine extracted Qsets into a list of type-consistent candidate equation trees (Section 3.4). Finally, ALGES uses discriminative models to score each candidate equation, using both local and global features (Section 3.5).

Specifically, the recursive nature of this representation permits me to decompose the likelihood function $p(t, w)$ into local scoring functions for each internal node of t followed by scoring the root node:

$$p(t|w) \propto \left(\prod_{t_j \in t} \mathcal{L}_{local}(t_j|w) \right) \times \mathcal{G}_{global}(t|w) \tag{3.1}$$

where the local function $\mathcal{L}_{local}(t_j|w)$ scores the likelihood of the subtree t_j, modeling pairwise Qset relationships, while the global function $\mathcal{G}_{global}(t|w)$ scores the likelihood of the root of t, modeling the equation in its entirety.

[4] Inspired by Semantically Augmented Parse Trees [Ge and Mooney, 2005] adapted to equational logic.

Learning. ALGES learns in a *weakly supervised* fashion, using word problems w_i and only their correct answer ℓ_i (not the corresponding equation tree) as training data $\{(w_i, \ell_i)\}_{i \in \{1,\ldots,N\}}$. I ground each w_i into ordered Qsets and generate a list of type-consistent candidate training equations T_{ℓ_i} that yield the correct answer ℓ_i.

I build a *local* discriminative model $\mathcal{L}_{local}$ to score the likelihood that a math operator $op \in \mathcal{O}$ can correctly combine two Qsets s_1 and s_2 based on their semantics and intertextual relationships. For example, in Figure 3.2 this model learns that $*$ has a high likelihood score for '7 buses' and 'x students'. The training data consists of feature vectors $\langle s_1, s_2 \rangle$ labeled with op, derived from the equation trees that yield the correct solution.

I also build a *global* discriminative model that scores equation trees based on the global problem structure: $\mathcal{G}_{global} = \psi^{\intercal} f_{global}(w, t)$ where f_{global} represents global features of w and t, and ϕ are parameters to be learned. The training data consists of feature vectors $\langle w, t \rangle$ for equation trees that yield the correct solution as positive examples, and the rest as negatives (Figure 3.2). The details of learning and inference steps are described in Section 3.5.

3.3 Grounding and Combining Qsets

I discuss how word problem text is grounded into an ordered list of Qsets. A Qset is a compact representation of important properties of a quantity which are local via dependency relations. The use of Qsets facilitates the building of semantically augmented equation trees. Additionally, by tracking certain properties of text quantities, ALGES can resolve pronominal references or elided nouns to properties of previous Qsets. It can also combine information about quantities referenced in different sentences into a single semantic structure for further use.

Grounding. ALGES translates the text of the problem w into interrelated base Qsets $\{s_1, \ldots, s_n\}$, each associated with a quantity in the problem text w. The properties of each Qset (Definition 1) are extracted from the Stanford English dependencies [De Marneffe and Manning, 2008] present in

For each quantity mentioned in the text, properties $(qnt, ent, ctr, adj, vrb, loc)$ of the corresponding Qset are extracted as follows:

1. qnt (quantity) is a numerical value or determiner found in the problem text, or a variable.
2. ent (entity) is a noun related to the qnt in the dependency parse tree. If qnt is a numerical value, ent is the noun related by the *num, number,* or *prep_of* relations. If qnt is a determiner, ent is the noun related via the *det* relation. When such a noun does not exist due to parse failure or pragmatic recoverability, ent is the noun that is closest to qnt in the same sentence or the ent associated with the most recent Qset.
3. ctr (container) is the subject of the verb governing ent, except in two cases: when this subject is a pronominal reference, the ctr is set to the ctr of the closest previous Qset; if ent is related to another Qset whose qnt is one of *each, every, a, an, per,* or *one*, ctr is set to the ent of that Qset.
4. adj (adjectives) is a list of adjectives related to ent by the *amod* relation.
5. vrb (verb) is a governing verb, either related to ent by *nsubj* or *dobj*
6. loc (location) is a noun related to *ent* by *prep_on, prep_in,* or *prep_at* relations.

Table 3.1: The process of forming a single Qset.

the sentence where the quantity is referred to according to the rules described in Table 3.1.[5]

Additionally, ALGES assigns a single target Qset s_x corresponding to the question sentence. The properties of the target Qset are also extracted according to the rules of the Table 3.1. In particular, the qnt property is set to unknown, the ent is set to the noun appearing after the words *what, many* or *much* in the target sentence, and the other properties are extracted as listed in Table 3.1.

Reordering. In order to reduce the space of possible equation trees, ALGES reorders Qsets $\{s_1, \ldots, s_n\}$ according to semantic and lexical information and enforces a constraint that Qsets can only combine with adjacent Qsets in the equation tree. In Figure 3.2, the target Qset corresponding to the unknown (x 'students') is moved from its textual location at the end of the problem and placed adjacent to the Qset with entity 'buses'. This move is triggered by the relationship between the target entity 'student' and its container 'bus' that is quantified by *each* in the last sentence. In addition to the container match rule, I employ three other rules to move the target Qset as described

[5]Dependencies are produced following the method of De Marneffe et al. [2006].

1. Move Qset s_i to immediately after Qset s_j if the container of s_i is the entity of s_j and is quantified by *each*.
2. Move target Qset to the front of the list if the question statement includes keywords *start* or *begin*.
3. Move target Qset to the end of the list if the question statement includes keywords *left, remain,* and *finish*.
4. Move target Qset to the textual location of an intermediate reference with the same *ent* if its *num* property is the determiner *some*.

Table 3.2: Rules for reordering Qsets.

in Table 3.2.[6]

Combining. Two Qsets and an arithmetic operator can be combined via the $combine$ function to form a third Qset, alternately referred to as a *compound*. Because of this, I can represent intermediate nodes in the equation tree as Qsets themselves. The recursive combination of Qsets allows me to effectively decompose equation trees into a collection of local operations over identical abstractions. This enables learning features of Qsets and text that indicate particular operations from both leaf and intermediate nodes. The mechanics of $c \leftarrow combine(a, b, op)$ are detailed below.

For $op = +$, the properties of either Qset a or b suffice to define c. ALGES always forms c using the properties of b in these situations. For $op = -$, the properties of the left operand a define the resultant set, as evidenced by the subtraction operations present in the first problem in Table 3.9 on page 51. To determine the stickers in Luke's possession, I need to track stickers related to the left Qset with the verb 'got'.

For $op = *$, the Qset relationship is captured by the container and entity properties: the one whose properties preserve after multiplication has the other's entity as its container. In Figure 3.2, the 'bus' Qset is the container of 'students'. When these are combined with the $*$ operator, the result is of entity type 'student'. For $op = /$, the type of the dividend is used as the type of the result.

[6]These reordering rules are intentionally minimal, but do provide some gain over both preserving the text ordering of quantities or setting ordering as a soft constraint. See Table 3.7.

3.4 Generating Equation Trees with ILP

ALGES uses an ILP optimization model to generate equation trees involving n base Qsets. These equation trees are used for both learning and inference steps. ALGES generates an ordered list of M of the most desirable candidate equations for a given word problem w using an ILP, which models global considerations such as type consistency and appropriate low expression complexity. To facilitate generation of equation trees, ALGESrepresents them in parenthesis-free *postfix* or *reverse Polish* notation, where a binary operator immediately follows the two operands it operates on (e.g., $abc{+}{*}x{=}$).

Given a word problem w with n base Qsets (cf. Table 3.3 for notation), I build an optimization model $\mathrm{ILP}(w)$ over the space of postfix equations $E = e_1 e_2 \ldots e_L$ of length L involving k numeric constants, $k' = n - k$ unknowns, r possible binary operators, and q "types" of Qsets, where type corresponds to the *entity* property of Qsets and determines which binary relationships are permitted between two given Qsets. For single variable equations over binary operators $\mathcal{O}$, $k' = 1, r = |\mathcal{O}| = 5$, and $L = 2n - 1$. For brevity, define $m = n + r$ and let $[j]$ denote $\{1, \ldots, j\}$. Expression E can be evaluated by considering $e_1, e_2, \ldots, e_L$ in order, pushing non-operator symbols on to a stack σ, and, for operator symbols, popping the top two elements of σ, applying the operator to them, and pushing the result back on to σ. The *stack depth* of the e_i is the stack size after e_i has been processed this way.

Variables. Integer variables $x_1, x_2, \ldots, x_L$ encode which symbol each e_i refers to. Their domain, $[m]$, represents the k numeric constants in the same order as their respective Qsets, followed by the k' unknowns, and finally operators in the order $+, -, *, /, =$. Binary variables c_i, u_i, and o_i indicate whether e_i is a numeric constant, unknown, or operator, resp. Variables d_i with domain $[L]$ equal the postfix stack depth of e_i. Finally, variables t_i with domain $[q]$ indicate the type of e_i. For $j \in [n]$, i.e., for the k constants and k' unknowns, $type_j \in [q]$ denotes the respective Qset entity. Uncertainty in object types may be incorporated easily by treating $type_j$ as a (potentially weighted) subset of $[q]$.

INPUT	
w	input math word problem
n	number of base Qsets
k	number of numeric constants
k'	number of unknowns (1 for single-var. eqns.)
r	number of binary operators ($r = \|\mathcal{O}\| = 5$)
m	number of possible symbols ($n + r$)
$type_j$	type of j-th base Qset
M	desired number of candidate equation trees
L	desired length of postfix equations ($2n - 1$)
OUTPUT	
E	postfix equation to be generated
e_i	i-th element of E; $i \in [L]$
VARIABLES for $i \in [L]$	
x_i	main ILP variable for i-th symbol of E
c_i	indicator variable: e_i is a numeric constant
u_i	indicator variable: e_i is an unknown
o_i	indicator variable: e_i is an operator
d_i	postfix stack depth of e_i; $d_i \in [L]$
t_i	type of e_i (corresponds to Qset entity); $t_i \in [q]$

Table 3.3: ILP notation for candidate equations model

Constraints and Objective Function. Constraints in $\mathrm{ILP}(w)$ include syntactic validity, type consistency, and domain specific simplicity considerations. The *objective function* minimizes the sum of the weights of violated soft constraints. Below, (H) denotes hard constraints, (W) weighted soft constraints, and (P) post-processing steps.

Definitional Constraints (H): Constraints over indicator variables c_i, u_i, and o_i ensure they represent their intended meaning, including the invariant $c_i + u_i + o_i = 1$. For stack depth variables, I add $d_1 = 1$ and $d_i = d_{i-1} - 2o_i + 1$ for $i > 1$.

Syntactic Validity (H): Validity of the postfix expression is enforced easily through constraints $o_1 = 0$ and $d_L = 1$. In addition, I add $x_L = m$ and $x_i < m$ for $i < L$ to ensure equality occurs exactly once and as the top-level operator.

Operand Access (H): The second operand of an operator symbol e_i is always e_{i-1}. Its first operand, however, is defined instead by the stack-based evaluation process. $\mathrm{ILP}(w)$ encodes it using an alternative characterization: the first operand of e_i is e_j iff $j \leq i - 2$ and j is the largest

index such that $d_i = d_j$.

Type Consistency (W): Suppose T_1 and T_2 are the types of the two operands of an operator o, whose type is T_o. Addition and subtraction preserve the type of their operands, i.e., if o is $+$ or $-$, then $T_o = T_1 = T_2$. Multiplication inherits the type of one of its operands, and division inherits the type of its first operand. In both cases, the two operands must be of different types. Formally, if o is $*$, then $T_o \in \{T_1, T_2\}$ and $T_1 \neq T_2$; if o is $/$, then $T_o = T_1 \neq T_2$.

Simplicity (H): I add a few hard simplicity constraints based on patterns observed in a small subset of the questions. These include an upper bound on the stack depth, which helps avoid overly complex expressions unsuitable for grade-school algebra, and reducing redundancy by, e.g., disallowing the numeric constant 0 to be an operand of $+$ or $-$ or the second operand of $/$.

Simplicity (W): To promote the use of simpler equations where possible, equations with a single variable alone on one side of the equality are preferred to equations with a single constant on one side of the equality, which are preferred to equations with complex expressions on both sides of the equality operator.

Symmetry Breaking (H): If a commutative operator is preceded by two numeric constants (e.g., $ab+$), I require the constants to respect their Qset ordering.

Symmetry Breaking (W): Every pair of constants that disrespects its Qset ordering incurs a small penalty, except those combined by a commutative operator.

Negative and Fractional Answers (P): Negative answers are out of scope for the word problems addressed in this work. However, imposing non-negativity in $\mathrm{ILP}(w)$ is unnecessarily complex. Instead, I filter out candidate expressions yielding a negative answer as a post-processing step. Similarly, when all numeric constants are integers, I filter out expressions yielding a fractional answer, again based on typical questions in these datasets.

3.5 Learning

My goal is to learn a scoring function that identifies the best equation tree t^* corresponding to an unseen word problem w. Since the dataset consists only of problem-solution pairs $\{(w_i, \ell_i)\}_{i=1,\ldots,N}$, training the scoring models requires producing equation trees matching ℓ_i. For every training instance (w_i, ℓ_i), $\text{ILP}(w_i)$ is used to generate M type-consistent equation tree candidates T_i. To train the local model (section 3.5.1), I filter out trees from T_i that do not evaluate to ℓ_i, extract all (s_1, s_2, op) triples from the remaining trees, and use feature vectors capturing (s_1, s_2) and labeled with op as training data (see Figure 3.2). For the global model, I use for training data a subset of T_i with an equal number of correct and incorrect equation trees (section 3.5.2). Once trained, I use Equation 3.1 to combine these models to compute a score for each candidate equation tree generated for an unseen word problem at inference time (see Figure 3.3).

3.5.1 Local Qset Relationship Model

I train a *local model* of a probability distribution over the math operators that may be used to $combine$ a pair of Qsets. The idea is to learn the correspondence between spans of texts and math operators by examining such texts and the Qsets of the involved operands. Given Qsets s_1 and s_2, the local scoring function scores the probability of each $op \in \{+, -, *, /\}$, i.e., $\mathcal{L}_{local} = \theta^\intercal f_{local}(s_1, s_2)$ where f_{local} is a feature vector for s_1 and s_2. Note that either Qset may be a compound (the result of a combine procedure). The goal is to learn parameters θ by maximizing the likelihood of the operators between every two Qsets that I observe in the training data. I model this as a multi-class SVM with an RBF kernel.

Features. Given the richness of the textual possibilities for indicating a math operation, the features are designed over semantic and intertextual relationships between Qsets, as well as domain-specific lexical features. The feature vector includes three main feature categories (Table 3.4).

First, *single set features* include syntactic and positional features of individual Qsets. For

example, they include indicator features for whether elements of a short lexicon of math-specific terms such as 'add' and 'times' appear near to the set reference in the surface string.[7] Also, following Hosseini et al. [2014], I include a vector that captures the distance between the verbs associated with each Qset and a small collection of verbs found to be useful in categorizing arithmetic operations in that work, based upon their Lin Similarity [Lin, 1998]. Second, *relationships between Qsets* are described w.r.t. various Qset properties described in section 3.2. These include binary features like whether one Qset's *container* property matches the other Qset's *entity* (a strong indicator of multiplication), or the distance between the verbs associated with each set based upon their Lin Similarity. Third, *target quantity features* check the matching between the target Qset and the current Qset as well as math keywords in the target sentence.

3.5.2 Global Equation Model

I also train a *global* model that scores equation trees based on the global structure of the tree and the problem text. The global model scores the compatibility of the tree with the soft constraints introduced in Section 3.4 as well as its correspondence with the problem text. I use a discriminative model: $\mathcal{G}_{global} = \psi^{\intercal} f_{global}(w, t)$ where f_{global} are the features capturing trees and their correspondences with the problem text. I train a global classifier to relate these features through parameters ψ.

Features f_{global} are explained in Table 3.4. They include the number of violated soft constraints in the ILP, the probabilities of the left and right subtrees of the root (representing the equality symbol) as provided by the local model, and global lexical features. Additionally, the three local feature sets are applied to the left and right Qsets.

[7] The use of surface distance for this lexicon is due to the difficulty of determining consistent dependency relations between the quantity and math term. This is often caused by parse errors arising from the math-specific versus general meaning of these terms.

1. Single Qset Features (repeated for B)
   - what argument of its governing verb is A?
   - is A a subset of another set?
   - is A a compound?
   - math keywords found in context of A
   - verb Lin distance from known verb categories (B only)

2. Relational features between Qsets A and B
   - entity match
   - adjective overlap
   - location match
   - distance in text
   - Lin similarity between verbs governing A and B
   - is one a subtype of the other?
   - does one contain the other?

3. Target Quantity features
   - A/B is target Qset
   - A/B entity matches target entity
   - math keywords in target context

---

4. Root node features
   - # of ILP constraints violated by equation
   - Scores of left and right subtrees of root

Figure 3.4: Features used for local and global models, for left Qset A and right Qset B

3.5.3 Inference

For an unseen problem w, I first extract base Qsets from w. The goal is to find the most likely equation tree with minimum violation of hard and soft constraints. Using $\mathrm{ILP}(w)$ over these Qsets, I generate M candidate equation trees ordered by the sum of the weights of the constraints they violate. I compute the likelihood score given by Eqn. (3.1) for each candidate equation tree t, use this as an estimate of the likelihood $p(t|w)$, and return the candidate tree t^* with the highest score. In Eqn. (3.1), the score of t is the product of the likelihood scores given by the local classifier for

each operand in t and the Qsets over which it operates, multiplied by the likelihood score given by the global classifier for the correctness of t. If the resulting equation provides the correct answer for w, I consider inference successful.

3.6 Experiments

This section reports on three experiments: a comparison of ALGES with Kushman et al. [2014]'s template-based method, a comparison of ALGES with Hosseini et al. [2014]'s verb-categorization methods, and ablation studies. The experiments are complicated by the fact that ALGES is limited to single equations, and the verb categorization method can only handle single-equations without multiplication or division. my main experimental result is to show an improvement over the template-based method on single-equation algebra word problems. I further show that the template-based method depends on lexical and template overlap between its training and test sets. When these overlaps are reduced, the method's accuracy drops sharply. In contrast, ALGES is quite robust to changes in lexical and template overlap (see Tables 3.4 and 3.5).

Experimental Setup. I use the Stanford Dependency Parser in CoreNLP 3.4 [De Marneffe et al., 2006] to obtain syntactic information used for grounding and feature computation. For the ILP model, I use CPLEX 12.6.1 [IBM ILOG, 2014] to generate the top $M = 100$ equation trees with a maximum stack depth of 10, aborting exploration upon hitting 10K feasible solutions or 30 seconds.[8] I use Python's SymPy package for solving equations for the unknown. For the local and global models, I use the LIBSVM package to train SVM classifiers [Chang and Lin, 2011] with RBF kernels that return likelihood estimates as the score.

Dataset. This work deals with grade-school algebra word problems that map to single equations with varying length. Every equation may involve multiple math operations including multiplication,

[8]These hyper-parameters were chosen based on experimentation with a small subset of the questions. A more systematic choice may improve overall performance.

Template Overlap	10.4	7.7	6.3	2.1
ALGES	**0.72**	**0.66**	**0.66**	**0.63**
Template-based	0.67	0.60	0.46	0.26
Error reduction	15%	15%	33%	50%

Table 3.4: Decreasing template overlap: Accuracy of ALGES versus the template-based method on single-equation algebra word problems. The first column corresponds to the SINGLEEQ dataset, and the other columns are for subsets with decreasing template overlap.

division, subtraction, and addition over non-negative rational numbers and one variable. The data is gathered from `http://math-aids.com`, `http://k5learning.com`, and `http://ixl.com` websites and a subset of the data from Kushman et al. [2014] that maps word problems to single equations. I refer to this dataset as SINGLEEQ (see Table 3.9 for example problems). The SINGLEEQ dataset consists of 508 problems, 1,117 sentences, and 15,292 words.

Baselines. I compare my method with the template-based method [Kushman et al., 2014] and the verb-categorization method [Hosseini et al., 2014] using the software provided by these researchers. For the template-based method, I use the fully supervised setting, providing equations for each training example.

3.6.1 Comparison with Template-based Method

I first compare ALGES with the template-based method over SINGLEEQ. I evaluate both systems on the number of correct answers provided and report the average of a 5-fold cross validation. ALGES achieves 72% accuracy whereas the template-based method achieves 67% accuracy, a 15% relative reduction in errors (first columns in Tables 3.4 and 3.5). This result is statistically significant with a p-value of 0.018 under a paired t-test.

Lexical Overlap. By further analyzing SINGLEEQ, I noted that there is substantial overlap between the content words (common noun, adjective, adverb, and verb lemmas) in different problems. For example, many problems ask for the total number of seashells collected by two people on a beach,

Lexical Overlap	4.3	3.3	2.6	2.5
ALGES	**0.72**	**0.66**	**0.66**	**0.63**
Template-based	0.67	0.60	0.46	0.26
Error reduction	15%	15%	33%	50%

Table 3.5: Decreasing lexical overlap: Accuracy of ALGES versus the template-based method on single-equation algebra word problems. The first column corresponds to the SINGLEEQ dataset, and the other columns are for subsets with decreasing lexical overlap.

with only the names of the people and the number of seashells that each found changed. To analyze the effect of this repetition on the learning methods evaluated, I define a *lexical overlap* parameter as the total number of content words in a dataset divided by the number of unique content words. The two "seashell problems" have a high lexical overlap.

Template Overlap. I also noted that many problems in SINGLEEQ can be solved using the same *template*, or equation tree structure above the leaf nodes. For example, a problem which corresponds to the equation $(9 * 3) + 7$ and a different problem that maps to $(4 * 5) + 2$ share the same template. I introduce a *template overlap* parameter defined as the average number of problems with the same template in a dataset.

Results. In my data, template overlap and lexical overlap co-vary. To demonstrate the brittleness of the template-based method simply, I picked three subsets of SINGLEEQ where both parameters were substantially lower than in SINGLEEQ and recorded the relative performance of the template-based method and of ALGES in Tables 3.4 and 3.5. The data used in both tables is the same, but the tables are separated for readability. The first column reports results for the SINGLEEQ dataset, and the other columns report results for the subsets with decreasing template and lexical overlaps. The subsets consist of 254, 127, and 63 questions respectively. I see that as the lexical overlap drops from 4.3 to 2.5 and as the template overlap drops from 10.4 to 2.1, the relative advantage of ALGES over the template methods goes up from 15% to 50%.

The template-based method's accuracy falls off significantly when faced with fewer repeated

templates or less spurious lexical overlap between problems (from 0.67 to 0.26) for algebra word problems with single equations. The accuracy of ALGES also declines from 0.72 to 0.63 across the table, which needs to be investigated further. In future work, I also need to investigate additional settings for the two parameters and to attempt to "break" their co-variance. Nevertheless, I have uncovered an important brittleness in the template-based method and have shown that ALGES is substantially more robust.

3.6.2 Comparison with Verb-Categorization

The verb-categorization method learns to solve addition and subtraction problems, while ALGES is capable of solving multiplication and division problems as well. I compare against their method over my dataset as well as the dataset provided by that work, here referred to as ADDSUB. ADDSUB consists of addition and subtraction word problems with the possibility of irrelevant distractor quantities in the problem text. The verb categorization method introduces a heuristic rule for handling irrelevant information: it ignores a quantity whose adjective is not consistent with the adjective of the target quantity. For example, if a problem describes sets of red balloons and sets of black balloons but the question asks specifically for the total number of black balloons, then quantities of red balloons are ignored under this rule. I augment ALGES with this rule for handling irrelevant information in ADDSUB.

Results, reported in Table 3.6, show comparable accuracy between both methods on Hosseini et al. [2014] data. My method shows a significant improvement versus theirs on the SINGLEEQ dataset due to the presence of multiplication and division operators, as 40% of the problems in my dataset include these operators.

Method	ADDSUB	SINGLEEQ
ALGES	0.77	**0.72**
Verb-categorization	0.78	0.48
Error reduction	-	53%

Table 3.6: Accuracy of ALGES compared to verb categorization method.

Method	Accuracy
ALGES	**0.72**
No Local Model	0.50
No Global Model	0.49
No Qset Reordering	0.68

Table 3.7: Ablation study of each component of ALGES.

3.6.3 Ablation Study

In order to determine the effect of various components of my system on its overall performance, I perform the following ablations:

No Local Model: Here, I test my method absent the local information (Section 3.5.1). That is, I generate equations using all ILP constraints, and score trees solely on information provided by the global model: $p(t|w) \propto \mathcal{G}_{global}(w, t)$.

No Global Model: Here, I test my method without the global information (Section 3.5.2). That is, I generate equations using only the hard constraints of ILP and score trees solely on information provided by the local model: $p(t|w) \propto \prod_{t_i \in t} \mathcal{L}_{local}(w, t_i)$.

No Qset Reordering: I test my method without the deterministic Qset reordering rules outlined in Section 3.3. Instead, I allow the ILP to choose the top M equations regardless of order.

Results in Table 3.7 show that each component of ALGES contributes to its overall performance on the SINGLEEQ corpus. I find that both the Global and Local models contribute significantly to the overall system, demonstrating the significance of a bottom-up approach to building equation trees.

Method	Accuracy
Local classifier: Full Feature set	**0.84**
No Single Set Features	0.81
No Set Relation Features	0.75
No Target Features	0.79

Table 3.8: Accuracy of local classifier in predicting the correct operator between two Qsets and ablating feature sets.

Importance of Features. I also evaluate the accuracy of the local Qset relationship model (Section 3.5.1) on the task of predicting the correct operator for a pair of Qsets $\langle s_1, s_2 \rangle$ over the SINGLEEQ dataset using a 5-fold cross validation. Table 3.8 shows the value of each feature group used in the local classifier, and thus the importance of details of the Qset representation.

3.6.4 Qualitative Examples and Error Analysis.

Table 3.9 shows some examples of problems solved by my method. I analyzed 72 errors made by ALGES on the SINGLEEQ dataset. Table 3.10 summarizes five major categories of errors.

Problems and equations
Luke had 20 stickers. He bought 12 stickers from a store in the mall and got 20 stickers for his birthday. Then Luke gave 5 of the stickers to his sister and used 8 to decorate a greeting card. How many stickers does Luke have left? $((20 + ((12 + 20) - 8)) - 5) = x$
Maggie bought 4 packs of red bouncy balls, 8 packs of yellow bouncy balls, and 4 packs of green bouncy balls. There were 10 bouncy balls in each package. How many bouncy balls did Maggie buy in all? $x = (((4 + 8) + 4) * 10)$
Sam had 79 dollars to spend on 9 books. After buying them he had 16 dollars. How much did each book cost? $79 = ((9 * x) + 16)$
Fred loves trading cards. He bought 2 packs of football cards for $2.73 each, a pack of Pokemon cards for $4.01, and a deck of baseball cards for $8.95. How much did Fred spend on cards? $((2 * 2.73) + (4.01 + 8.95)) = x$

Table 3.9: Examples of problems solved by ALGES together with the returned equation.

Error type	Example
Parsing Issues (12%)	Randy needs 53 cupcakes for a birthday party. He already has 7 chocolate cupcakes and 19 vanilla cupcakes. How many more cupcakes should Randy buy?
Grounding & Ordering (19%)	There are 24 bicycles and 14 tricycles in the storage at Danny's apartment building. Each bicycle has 2 wheels and each tricycle has 3 wheels. How many wheels are there in all?
Semantic Limitation (19%)	The sum of three consecutive even numbers is 162. What is the smallest of these numbers?
Lack of Knowledge (32%)	A restaurant sold 63 hamburgers last week. How many hamburgers on average were sold each day?
Inferring quantities (18%)	Sara, Keith, Benny, and Alyssa each have 96 baseball cards. How many dozen baseball cards do they have in all?

Table 3.10: Examples of different error categories and relative frequencies. Sources of errors are underlined.

Parsing errors cause a wrong grounding into the designed representation. For example, the parser treats 'vanilla' as a noun modified by the number '19', leading my system to treat 'vanilla' as the entity of a Qset rather than 'cupcake'. Despite the improvements that come from ALGES, a portion of errors are attributed to grounding and ordering issues. For instance, the system fails to correctly distinguish between the sets of wheels, and so does not get the movement-triggering container relationships right. Semantic limitations are another source of errors. For example, ALGES does not model the semantics of 'three consecutive numbers'. The fourth category refers to errors caused due to lack of world knowledge (e.g., 'week' corresponds to '7 days'). Finally, ALGES is not able to infer quantities when they are not explicitly mentioned in the text. For example, the number of containers of baseball cards (who in this case are people) should be inferred by understanding that the set quantified by 'each' consists of these 4 people.

3.7 Conclusion

In this work I have outlined a method for solving grade school algebra word problems. I have empirically demonstrated the value of this approach versus state-of-the-art word problem solving techniques. My method grounds quantity references, utilizes type-consistency constraints to prune the search space, learns which algebraic operators are indicated by text, and ranks equations according to a global objective function. ALGES is a hybrid of previous template-based and verb categorization state-based methods for solving such problems. By learning correspondences between text and mathematical operators, I extend the method of state updates based on verb categories. By learning to re-rank equation trees using a global likelihood model, I extend the method of mapping word problems to equation templates. ALGES achieves state-of-the-art results on the SINGLEEQ dataset, and the robustness of the method is demonstrated on data with lower template and lexical overlap.

Future work involves extending ALGES to solve higher grade math word problems including simultaneous equations. This can be accomplished by extending the variable grounding step to allow multiple variables, and training the global equation model to recognize which quantities belong to which equation.

Chapter 4

Generating Longer Texts by Editing

Storytelling is the complex activity of expressing a plot, its events and participants in words meaningful to an audience.[1] Automatic storytelling systems can be used for customized sport commentaries, enriching video games with personalized or dynamic plot-lines [Barros and Musse, 2007], or providing customized learning materials which meet each individual student's needs and interests [Bartlett, 2004]. In this paper, I focus on generating narrative-style math word problems (Figure 4.1). Building on what I discovered about the math word problem genre in Chapter 3, I demonstrate that it is possible to design an algorithm that can automatically change the overall theme of a text without changing its correspondence to an underlying equation. This model can be used to create more engaging homework that is related, for example, to the theme of a student's favorite movie.

A math word problem is a coherent story that provides the student with good clues to the correct mathematical operations between the numerical quantities described therein. However, the particular *theme* of a problem, whether it be about collecting apples or traveling distances through

[1]The work presented in this chapter was done with Ioannis Konstas, Luke Zettlemoyer, and Hannaneh Hajishirzi. It appears as Koncel-Kedziorski et al. [2016a]

Craig walked 0.2 of a mile from school to David's house and 0.7 of a mile from David's house to his own house. How many miles did Craig walk in all?
Star Wars Uncle Owen walked 0.2 of a mile from hangar to Luke Skywalker's room and 0.7 of a mile from Luke Skywalker's room to his own room. How many miles did Uncle Owen walk in all?
Cartoon Finn squished 0.2 of a mile from cupboard to Melissa's dock and 0.7 of a mile from Melissa's dock to his own dock. How many miles did Finn squish in all?
Western Duane strolled 0.2 of a mile from barn to Madeline's camp and 0.7 of a mile from Madeline's camp to his own camp. How many miles did Duane stroll in all?

Figure 4.1: An example story and rewrites in 3 themes.

space, can vary significantly so long as the correlation between the story and underlying equation is maintained. Students' success at solving a word problem is tied to their interest in the problem's theme [Renninger et al., 2002], and personalizing word problems increases student understanding, engagement, and performance in the problem solving process [Hart, 1996; Davis-Dorsey et al., 1991].

Motivated by this need for thematically diverse, highly coherent stories, I address the problem of *story rewriting*, or transforming human-authored stories into novel, coherent stories in a new theme. Rather than synthesizing first a story plot [McIntyre and Lapata, 2009, 2010] or script [Chambers and Jurafsky, 2009; Pichotta and Mooney, 2016; Granroth-Wilding and Clark, 2016] from scratch, I instead begin from an existing story and iteratively edit it towards a thematically novel but—most crucially—semantically compatible story. This approach allows me to reuse much, but not all, of the syntactic and semantic structure of the original text, resulting in the creation of more coherent and solvable math word problems.

I define a theme to be a collection of reference texts, such as a movie script of series of books. Given a theme, the *rewrite* algorithm constructs new texts by substituting thematically appropriate words and phrases, as measured with automatic metrics over the theme text collection, for parts of the original texts. This process optimizes for a number of metrics of overall text quality, including

syntactic, semantics, and discourse scores. It uses no hand crafted templates and requires no theme-specific tuning data, making it easy to apply for new themes in practice.[2] Tables 4.4–4.6 show example stories generated from the rewrite system.

To evaluate performance, I collected a corpus of 450 rewrites of math word problems in Star Wars and Children's Cartoon themes via crowdsourcing. Experiments with automated metrics and human evaluations demonstrate that the approach described here outperforms a number of baselines and can produce solvable problems multiple different themes, even with no in-domain tuning.

4.1 Problem Formulation

My system takes as input a story s and a theme t, and outputs the best rewrite s^* from generated candidates S.

A theme t is defined as a textual corpus that describes a topic or a domain. Here, I use a flexible notion of a theme and allow a variety of text to serve as unique themes. For example, a theme can be the collection of all Science Fiction stories from the Project Gutenberg, or the movie script for a single horror movie, or a collection of fan fiction from the Internet.

The generated candidate s^* is the most *thematically* fit problem that is *syntactically* and *semantically* coherent given the original problem s and the new theme t. I represent a story in terms of the words it contains, so that $s = \{w_1, w_2, \dots\ w_n\}$ and $|s| = n$. The new story s' is defined as:

$$s' = \big\{f(w_1), f(w_2), \dots\ f(w_n)\big\}$$

where the function $f(w) : \mathcal{V}_o \rightarrow \mathcal{V}_t^K \cup \varnothing$, rewrites a word from the vocabulary of the original problem $\mathcal{V}_o$ to either a word, a trivial noun compound of length K (e.g., multi-word named entity) from the vocabulary of the the thematic vocabulary $\mathcal{V}_t$, or reduces to the empty symbol, i.e., omits the input word entirely; hence the length of s' can differ from that of the original problem.

[2] My method could also be viewed as constructing templates on-the-fly from each original text.

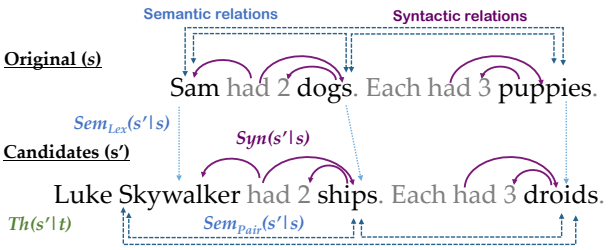

Figure 4.2: An overview of my method for scoring a candidate story s' given a human-authored story s and a theme t. Syn$(s'|s)$: compatiblity of syntactic relations (purple arrows), Sem$_{pair}(s'|s)$: coherence of semantic relations (blue arrows), Sem$_{Lex}(s'|s)$: semantic mapping of individual words, and $Th(s'|s,t)$: thematicity.

Formally, my goal is to select the candidate $s' \in S$ by maximizing a scoring function $\mathcal{R}$ over thematic, syntactic and semantic constraints, subject to a set of parameters θ:

$$s^* = \arg\max_{s' \in S} \mathcal{R}(s'|s,t;\theta) \tag{4.1}$$

In order to find the best story s^*, my problem reduces to generating candidate stories s' from the space of possible rewrites of the human-authored story s in a new theme t (Section 4.3). Since there are exponentially many rewrites, I follow a greedy two-stage decoding approach: first I identify only the content words w_i in the input problem, and provide for each a list of the top-k most *salient* thematic candidate words, and trivial noun compounds. I then search the space by progressively introducing more rewrites in the beam, and scoring them according to $\mathcal{R}$ (Section 4.2). Figure 4.2 shows the overview of the scoring function for a candidate sentence s'.

4.2 Scoring Stories

The scoring function $\mathcal{R}$ decomposes into three components, capturing aspects of syntactic compatibility, semantic coherence, and thematicity:

$$\begin{aligned}\mathcal{R}(s'|s,t;\theta) =& \alpha \times \text{Sem}(s'|s) \\ &+ \beta \times \text{Syn}(s'|s) \\ &+ \gamma \times \text{Th}(s'|s,t)\end{aligned} \tag{4.2}$$

The syntactic (Syn) and semantic (Sem) coherence components measure the coherence of the words in the new story s', as well as their compatibility to the syntactic and semantic relations in the original story s. On the other hand, thematicity (Th) scores the relevance and importance of words in the new story with respect to theme t.

I describe each of these components and the decoding process in the following sections.

4.2.1 Thematicity

Recall that a theme t is defined as a collection of documents that share a common topic, such as books in the science fiction genre, or scripts of horror movies. I define thematicity of a word w'[3] as the measure of *salience*, or how discriminative that word is to a given theme. For example, *robot* and *spaceship* are expected to be highly thematic with respect to Star Wars. In my setting I extend this definition to a candidate problem s' given s and t as:

$$\text{Th}(s'|s,t) = \sum_{i}^{|s|} Sal(w'_i, t) \tag{4.3}$$

[3]I will be interchangeably referring to w' as either the word or the head of the multi-word noun compound that rewrites the equivalent word w in the original problem.

where w'_i is a word from the candidate problem, and Sal is its salience score with respect to the theme. In the context of this work I argue that the thematic adaptation of the content words, i.e., nouns, verbs, named entities, and adjectives, plays the most important role in forming a new thematic problem. Therefore, I define their salience (except named entities) based on their tf-idf score over the theme t, and set it to zero for function words as defined by an online resource [SequencePublishing, 2016]. Since thematically central named entities can have low tf-idf (e.g. 'Luke Skywalker' is often referred to as 'Luke'), I set their salience to $1 - \frac{1}{c(w'_i)}$, where $c(w'_i)$ is the number of times w'_i occurs in the theme. In the example story in Figure 4.2 the thematicity score is derived as $Sal(\text{Luke Skywalker}) + Sal(\text{ships}) + Sal(\text{droids})$.

4.2.2 Syntactic compatibility

This work offers a new method for syntactic and discourse coherence based on preserving human-authored syntactic structure in generated text (hence my use of the term *rewriting*). The syntactic constructs in a document play a distinctive role in maintaining a cohesive discourse narrative across sentences. I consider the human-authored syntax of the original story s as gold standard, and use it to score a candidate problem s' by considering how well the syntactic relations of s apply to s'.

Formally, given a dependency triple (w_i, w_j, l) from a parse of a sentence in s, I compute the likelihood for the corresponding triple (w'_i, w'_j, l) for w'_i, w'_j in s'. I define the syntactic score for all sentences in s' as:

$$\text{Syn}(s'|s) = \sum_{i,j,l|(w_i,w_j,l)\in \text{Dep}(s)} \mathcal{L}_{Dep}(w'_i, w'_j, l) \tag{4.4}$$

where $Dep(s)$ are the dependency parse trees[4] for all sentences in s; $\mathcal{L}_{Dep}$ is a 3-gram language model, over dependency triples, i.e., corresponds to the likelihood of an arc label l being used between a pair of words (w'_i, w'_j). For example in Figure 4.2, the syntactic compatiblity score includes dependency likelihoods of $\mathcal{L}_{Dep}(\text{ship}, 2, \text{num})$, $\mathcal{L}_{Dep}(\text{had}, \text{ship}, \text{dobj})$.

[4]as provided by Manning et al. [2014]

Therefore, the Syn function prefers stories s' that (a) have similar dependency structure to the original story s and (b) make use of a common syntactic configuration.

4.2.3 Semantic Coherence

The semantic coherence component expresses how well a candidate s' rewrites individual words and realizes the semantic relationships that exist in the human-authored story s. Ideally, I would like to preserve enough of the semantics of s in order to produce a coherent story s', yet I am populating s' with words taken from an unrelated theme. Therefore, I model the semantics of a story s' in terms of the lexical semantics contributed by individual words as well as semantic relationships that exist between its elements. Note that the relationships can cross the sentence boundaries, promoting discourse coherence.

I decompose semantic relations in a story into a set of local, lexical relationships between pairs of words. More specifically, I consider semantic relations for noun-noun and verb-verb pairs as provided by WordNet [Miller, 1995]. Since some relations are not directly outlined in these resources (e.g., the selectional preferences of nouns with regard to their adjectival modifiers), I also consider the word-embedding similarity between words. For example in Figure 4.2 the semantic relationships are denoted with blue arrows between pairs of content words in the story (e.g., {Sam, dogs}, {dogs, puppies}, etc).

More formally, I define the semantic coherence of s' with respect to s as:

$$\begin{aligned} \text{Sem}(s'|s) = &\sum_{i}^{|s'|} \text{Sem}_{Lex}(w_i, w'_i) \\ &+ \sum_{i,j \in CW} \text{Sem}_{Pair}(\{w_i, w_j\}, \{w'_i, w'_j\}) \end{aligned} \tag{4.5}$$

where CW is the set of pairs of indices of content words (nouns, verbs, adjectives, and named entities) from s. I focus on the content words of the original problem, as they carry most of the

semantic information. Sem_{Lex} and Sem_{Pair} functions are semantic adaptation scores for individual words and semantic relations respectively, described below.

Semantic Compatibility between words (Sem_{Lex}) is defined as:

$$\text{Sem}_{Lex}(w_i, w_i') = cos(w_i, w_i') + Resnik(w_i, w_i') \tag{4.6}$$

where $cos(w_i, w_i')$ denotes the cosine similarity between the vector space embeddings of two words w_i and w_i'[5], and $Resnik(w_i, w_i')$ expresses the information content of the lowest subsumer of $\{w_i, w_i'\}$ in WordNet. For example in Figure 4.2, the semantic compatibility score incorporates lexical similarities $\text{Sem}_{Lex}(\text{dog}, \text{ship})$, etc.

Compatibility score between semantic relations (Sem_{Pair}) is defined by adding two components: $PairSim$ and $Analogy$ that compute how semantic relations between pairs of words are preserved in the new story:

$$PairSim = \cos(w_i, w_j) * \cos(w_i', w_j') \tag{4.7}$$

$$Analogy = \cos(w_i' + w_j - w_i, w_j') \tag{4.8}$$

$PairSim$ preserves the similarity between pairs of words $\{w_i, w_j\}$ in s and the corresponding pair $\{w_i', w_j'\}$ in the new story s'. Intuitively, if w_i and w_j are semantically close to each other, I would like the corresponding words to be close in the new story as well. For example in Figure 4.2, 'dog' and 'puppy' are similar in the original story, I expect the corresponding words 'ship' and 'droid' to be similar in the new story. The $Analogy$ function, inspired by Mikolov et al. [2013], computes the analogy of w_j' from w_i' given the relationship that holds between w_i and w_j in the vector space. For example in Figure 4.2, the relation between 'Sam' to 'dog' is similar to the relation between 'Luke Skywalker' to 'ship'.

[5]For the ease of notation, I represent the embedding of the words with w_i as well.

4.3 Decoding

My decoding process begins by first identifying the content words w_i (nouns, verbs, adjectives and named entities as provided by Manning et al. [2014]) in the original problem s that will be considered as *initial points* for rewriting. For each of these lexical classes I extract the top-k most thematic words and trivial noun compounds from the theme t. For example, in Figure 4.2, candidate nouns are: 'ships', 'robots', 'droids', etc., and for verbs: 'blast', 'soar', 'command', etc. Recall that the space of candidate rewrites is large, prohibiting an exhaustive enumeration. I therefore perform greedy search with a beam, by considering simultaneously all possible *paths* that start at the different initial points. At each step the decoder considers an additional rewrite from the list of candidates, adds it to the existing hypothesis path, and scores it according to function $\mathcal{R}$ (Equation 4.2).

All the counterpart scores are locally optimal, as they factor over each new word w'_i or pair of $\{w'_i, w'_j\}$, where w'_j is a rewrite *already* existing in the hypothesis path. At any given step I may recombine hypotheses that share the same prefix hypothesis path, and keep the top scoring one. The process terminates when there are no more rewrites left. I also experimented decoding with a variety of orderings of the text in the original problem s, including left-to-right, and head-first following the dependency tree of each sentence and then concatenating these linearizations; I observed that considering multiple paths achieves the best performance.

4.4 Data Collection

For the set of human-authored stories $\{s\}$, I use a corpus of math word problems described in Koncel-Kedziorski et al. [2016b]. This dataset extends the SINGLEEQ dataset described in Chapter 3. It consists of 3,221 math word problems with equation annotations and amalgamates several popular datasets used in previous works. I select a random subset of the problems targeting 5th and 6th grade levels which involve a single equation in one variable, totalling 150 problems. These problems

have 2.7 sentences and 29.4 words on average, 12.6 of which are considered content words by my system.

In order to tune and evaluate my model, I then collect a corpus of human-authored rewrites produced by workers from Amazon Mechanical Turk, based on two themes, i.e., Star Wars, and Children's Cartoon. I experimented with different ways of helping to define the theme for the crowd workers, including offering automatically generated word clouds or enforcing that a response includes one of several keywords. In practice, I have found that using specific cultural elements as themes (such as famous movie or cartoon franchises) attracts workers who already have a strong knowledge of the theme, resulting in higher quality work.

To help explain the rewriting process, I show workers three examples of thematic rewrites with varying degrees of correlation to the original problems. I then show workers a random problem from the original set $\{s\}$ and a corresponding equation for that problem. I instruct the workers to "rewrite" the problem according to the theme, ensuring that their rewritten problem can be solved by the provided equation.

I first collected Star Wars themed rewrites from workers with the "master" designation and at least 95% approval rating to ensure quality responses. This dataset, SWDEV , is used primarily for parameter tuning. I then manually identified high-performing workers by examining these rewrites and asked them if they wanted to complete more tasks. The SWTEST and CARTOON datasets, used for testing only, were written by this subset of high-performing authors of SWDEV who self-identify as theme experts. Each of SWDEV , SWTEST , and CARTOON consist of 3 rewrites for each of 50 non-overlapping original problems. The final dataset collection comprises of 450 human-authored rewrites.

4.5 Experiments

4.5.1 Setup

Implementation Details I pre-process the themes using the Stanford CoreNLP tools [Manning et al., 2014] for tokenization, Named Entity Recognition [Finkel et al., 2005], POS tagging [Toutanova et al., 2003], and dependency parsing [Chen and Manning, 2014]. For calculating salience scores, I use the ScriptBase dataset of movie scripts [Gorinski and Lapata, 2015]. The Star Wars theme is constructed from the ScriptBase version of 'Star Wars Episode IV' script, roughly 7300 words. The Cartoon theme is constructed from fan-authored scripts of the first 10 episodes of the show [Springfield, 2016] totalling 1370 words. To filter potentially offensive content, I prohibit the use of words and phrases from a list of offensive terms published by The Racial Slur database [Racial Slur Database, 2016] and FrontGate Media [Media, 2016]. To prohibit overgeneration, I prohibit the transformation of 1000 very common words, function words [SequencePublishing, 2016], or math-specific words [Survivors, 2013; Koncel-Kedziorski et al., 2015].[6]

For syntactic compatiblity score Syn (Equation 4.4) I use the English Fiction subset of the Google Syntactic N-grams corpus [Goldberg and Orwant, 2013] and train a 3-gram language model using KenLM [Heafield, 2011]. For Sem_{Lex}, $PairSim$ and $Analogy$ (Equations 4.6-4.8) I use the pretrained word embeddings of Levy and Goldberg [2014]. These embeddings are trained using dependency contexts rather than windows of adjacent words, allowing them to capture functional word similarity. Finally, I tune the parameters of my model (Equation 4.2) on the development set SWDEV and pick those values[7] that maximize METEOR score [Denkowski and Lavie, 2014] against 3 human references.

[6]The prohibited word lists are available at `https://gitlab.cs.washington.edu/kedzior/Rewriter/tree/master/data`

[7]I set $\alpha = 0.1$, $\beta = 0.1$ and $\gamma = 1$

Model	SWDEV	SWTEST	CARTOON
FULL	31.82	29.16	32.08
-SEM	28.72	25.55	27.55
-SYN	31.92	29.14	32.04

Table 4.1: METEOR results for different configuration of my model on SWDEV, SWTEST and CARTOON datasets.

Evaluation I compare two ablated configurations of my method against my full model (FULL): -SYN that only uses semantic and thematicity components and does not incorporate the syntactic compatibility score, -SEM replaces the semantic coherence score with the simpler $\cos(w_i, w_i')$, effectively rewriting only single words, and not pairs. I refrained from ablating the thematicity score as it is the core part of my model that drives the rewriting process into a new theme.

I evaluate my method using an automatic metric, and via eliciting human judgments on Amazon Mechanical Turk. For automatic evaluation, I compute the METEOR score, comparing the output of each model for a given problem and theme to the 3 human rewrites I collected, on SWDEV, SWTEST and CARTOON. METEOR is a recall-oriented metric, widely used in the MT community, which includes stemming, synonyms, and paraphrase matching in its scores.[8]

For human evaluation, I first conduct two pairwise comparison tests, pairing FULL against a human rewrite (HUMAN), and FULL against -SYN. Participants were given a short description of the theme, and the output of each system. For each test I asked 50 subjects to select which problem they preferred over 10 pairs of outputs; I obtained a total of 50 (5x10) responses for SWTEST and CARTOON.

In order to understand more fine-grained properties of the generated stories, I conduct a detailed human evaluation. I present 8 participants with the output of the three automatic systems, human rewrites (HUMAN), and a theme. The participants were asked to rate the stories across three dimensions: coherence (how coherent is the text of the problem?), solvability (can elementary school students solve it?), and thematicity (how well does the problem express them?) on a scale

[8]The average METEOR score comparing 1 annotator against the other 2 is 0.26, indicating that there are diverse correct strategies for solving the rewriting problem.

Model	SWTEST	CARTOON
FULL	65.0	57.9
-SYN	35.0	42.1
FULL	17.9	10.0
HUMAN	82.1	90.0

Table 4.2: Human evaluation results on pairwise comparisons between FULL and -SYN, and FULL and HUMAN, on SWTEST and CARTOON datasets.

Model	Thematicity	Coherence	Solvability
HUMAN	3.7	3.175	4.025
FULL	3.7	3.025	3.9
-SYN	3.375	3.075	3.825
-SEM	3.325	2.65	3.7

Table 4.3: Human evaluation results for FULL, -SYN, -SEMand HUMAN on thematicity, coherence and solvability on SWTEST.

from 1 to 5. I collected ratings over 16 outputs from SWTEST. Quality assurance questions were included to prevent random guessing, and responses which did not correctly answer these questions were disregarded, resulting in 128 responses.

4.5.2 Results

Table 4.1 reports METEOR; I notice that removing the semantic coherence scores in -SEM hurts the performance compared to FULL; this confirms my claim that semantic compatibility is crucial for building coherent stories. On the other hand, -SYN performs similarly to FULL. Closer inspection of the -SYN system's output reveals a greater diversity in thematic elements, as a result of the relaxed syntactic compatibility constraints. Hence it is more likely to have greater overlap with any of the reference rewrites, and rewarding it more positively.

I investigate this by performing a pairwise comparison between FULL and -SYN (Table 4.2); I observe that human subjects consistently prefer the output of FULL instead of -SYN both for SWTEST and CARTOON. Table 4.2 also reports that HUMAN outperforms the output of the FULL model. Table 4.3 shows the results of the detailed comparison of Thematicity, Coherence, and

Star Wars
s_1. Wendy bought 4 new chairs and 4 new tables for her house. If she spent 6 minutes on each piece furniture putting it together, how may minutes did it take her to finish?
s'_1. Leia bought 4 new ships and 4 new guns for her room. If she spent 6 minutes on each wasteland weapon putting it together, how many minutes did it take her to terminate?
s_2. My car gets 20 miles per gallon of gas. How many miles can I drive on 5 gallons of gas?
s'_2. My cruiser gets 20 miles per gallon of light. How many miles can I drive on 5 gallons of light?
s_3.Tyler had 15 dogs. Each dog had 5 puppies. How many puppies does Tyler now have?
s'_3. Biggs had 15 creatures. Each creature had 5 creatures. How many creatures does Biggs now have?

Table 4.4: Examples of the original stories s_i and rewritten math word problems s'_i in Star War theme.

Solvability. This table clearly shows the strong contribution of the semantic component of my system. The specific contribution of the syntactic component is to produce overall more solvable and thematically satisfying problems, although it can slightly affect coherence especially when automatic parses fail. Finally, the overall high ratings for human-authored stories across all three dimensions confirm the high quality of the crowd-sourced stories.

4.5.3 Qualitative Examples

Table 4.4–4.6 shows some problems generated by my method. Recall that since my system needs no annotated thematic training data, I can easily generate from any theme where thematic text is available. To demonstrate this fact, I include generated examples in a Western theme from novels from the Project Gutenberg corpus. Many of the results of my system are very legible, with only minor agreement errors. Coherent, thematic semantic relations are evident in problems such as s'_1, where ships, guns, and weapons combine to effect the Star Wars theme; this is also evident in s'_5, where people with western sounding names like Kurt and Madeline trade in cigarettes, an old-fashioned pre-cursor to e-cigarettes.

In some cases, semantic inconsistencies result in weird sounding problems, such as in s'_6 where the main character receives "wheat of grub". My model still scores this candidate highly due to the

Cartoon
s_7. Dave was helping the cafeteria workers pick up lunch trays, but he could only carry 9 trays at a time. If he had to pick up 17 trays from one table and 55 trays from another, how many trips will he make? s'_7. Finn was helping the cupboard men pick up candy bottles, but he could only carry 9 bottles at a time. If he had to pick up 17 bottles from one ring and 55 bottles from another, how many swords will he make?
s_8. If books came from all the 4 continents that Bryan had been into and he collected 122 books per continent, how many books does he have from all 4 continents combined? Finn had been into and he collected 122 dances per mountain, how many dances does he have from all 4 mountains combined?
s_9. A bucket contains 3 gallons of water. If Derek adds 6.8 gallons more, how many gallons will there be in all? s'_9. A bottle makes 3 gallons of serum. If Finn adds 6.8 gallons more, how many gallons will there be in all?

Table 4.5: Examples of the original stories s_i and rewritten math word problems s'_i in Cartoon theme.

strong semantic connection between "wheat" and "graze".

Semantic incoherence is less of a problem in the cartoon theme, where absurd interactions between characters are expected. However, a difficulty for my system is demonstrated in s'_7, where the physical entity "swords" is substituted for the nominalization of an event "trips". A semantic coherence component which could identify the cardinality of the set quantified by 'each' could resolve such issues.

4.6 Conclusion

I formalized the problem of story rewriting as automatically changing the theme of a text without altering the underlying story and developed an approach for rewriting algebra word problems, where the rewriting model optimized for a number of measures of overall text coherence. Experiments on a newly gathered dataset demonstrated my model can produce on theme texts that are usually solvable.

In future work, I plan to improve the thematicity and solvability components by incorporating

Western
s_4. Christians father and the senior ranger gathered firewood as they walked towards the lake in the park and brought with them sacks...
s'_4. Christian 's partner and the lone sheriff harvested barley as they strolled towards the hip in the orchard and plucked with them bags...
s_5. Sally had 27 cards. Dan gave her 41 new cards. Sally bought 20 cards. How many cards does Sally have now?
s'_5. Madeline had 27 cigarettes. Kurt gave her 41 new cigarettes. Madeline bought 20 cigarettes. How many cigarettes does Madeline have now?
s_6. For Halloween Megan received 11 pieces of candy from neighbors and 5 pieces from her older sister. If she only ate 8 pieces a day, how long would the candy last her?
s'_6. For Halloween Madeline received 11 wheat of grub from proprietors and 5 wheat from her nameless partner. If she only grazed 8 wheat a day, how long would the grub last her?

Table 4.6: Examples of the original stories s_i and rewritten math word problems s'_i in Western theme.

domain-specific and commonsense knowledge, leveraging information extraction. I also plan to study rewriting in other domains such as children short stories and extend the model to generate math word problems directly from equations. Finally, I intend to incorporate the generated problems in educational technology and tutoring systems.

Chapter 5

Knowledge Graphs as Document Plans

Chapter 4 described an editing method for generating novel texts, but this method has several shortcomings. The thematic, syntactic, and semantic constraints interacted only weakly though the fixed hyperparameters. In many instances the syntactic constraints were too strong; the model's inability to revise syntax resulted in ungrammatical outputs. Additionally, the semantic coherence measures were drawn from unsupervised (word2vec) or unrelated (WordNet) data. None of these sources consider words in multi-sentence contexts.

The biggest problem with the rewriter model presented above is the reliance on close neighbor texts that can be edited into the target text. If such a neighbor does not exist (as is often the case with longer texts), the editing method may not be best. What would be better is a model that can generate topical multi-sentence texts from scratch.

Recent increases in computing power and model capacity have made it possible to generate mostly-grammatical sentence-length strings of natural language text.[1] However, generating several sentences related to a topic and which display overall coherence and discourse-relatedness is an open challenge. The difficulties are compounded in domains of interest such as scientific writing. Here the variety of possible topics is great (e.g. topics as diverse as driving, writing poetry, and

[1]The work presented in this chapter was done with Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. It appears as Koncel-Kedziorski et al. [2019].

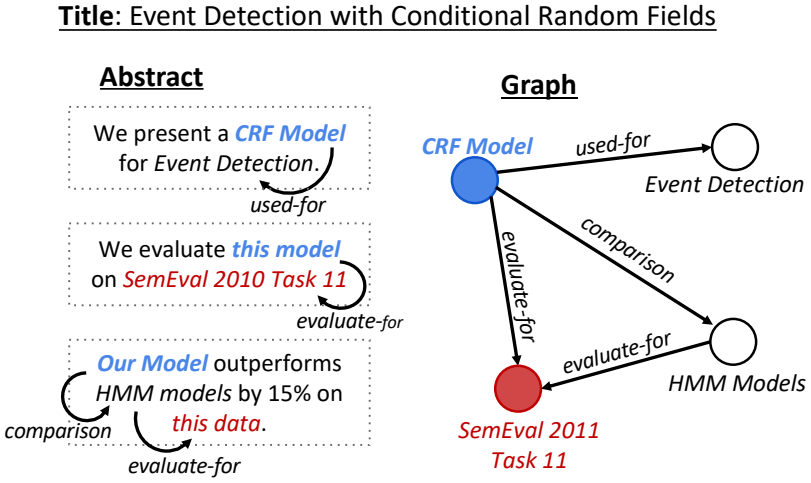

Figure 5.1: A scientific text showing the annotations of an information extraction system and the corresponding graphical representation. Coreference annotations shown in color. My model learns to generate texts from automatically extracted knowledge using a graph encoder decoder setup.

picking stocks are all referenced in one subfield of one scientific discipline). Additionally, there are strong constraints on document structure, as scientific communication requires carefully ordered explanations of processes and phenomena.

Many researchers have sought to address these issues by working with structured inputs. Data-to-text generation models [Konstas and Lapata, 2013; Lebret et al., 2016; Wiseman et al., 2017; Puduppully et al., 2019] condition text generation on table-structured inputs. Tabular input representations provide more guidance for producing longer texts, but are only available for limited domains as they are assembled at great expense by manual annotation processes.

The current work explores the possibility of using information extraction (IE) systems to automatically provide context for generating longer texts (Figure 5.1). Robust IE systems are available and have support over a large variety of textual domains, and often provide rich annotations of relationships that extend beyond the scope of a single sentence. But due to their automatic nature, they also introduce challenges for generation such as erroneous annotations, structural variety, and significant abstraction of surface textual features (such as grammatical relations or predicate-

argument structure).

To effect my study, I use a collection of abstracts from a corpus of scientific articles [Ammar et al., 2018]. I extract entity, coreference, and relation annotations for each abstract with a state-of-the-art information extraction system [Luan et al., 2018], and represent the annotations as a *knowledge graph* which collapses co-referential entities. An example of a text and graph are shown in Figure 5.1. I use these graph/text pairs to train a novel attention-based encoder-decoder model for knowledge-graph-to-text generation. My model, GraphWriter, extends the successful Transformer for text encoding [Vaswani et al., 2017] to graph-structured inputs, building on the recent Graph Attention Network architecture [Veličković et al., 2018]. The result is a powerful, general model for graph encoding which can incorporate global structural information when contextualizing vertices in their local neighborhoods.

The main contributions of this work include:

1. I propose a new graph transformer encoder that applies the successful sequence transformer to graph structured inputs.

2. I show how IE output can be formed as a connected unlabeled graph for use in attention-based encoders.

3. I provide a large dataset of knowledge-graphs paired with scientific texts for further study.

Through detailed automatic and human evaluations, I demonstrate that automatically extracted knowledge can be used for multi-sentence text generation. I further show that structuring and encoding this knowledge as a graph leads to improved generation performance compared to other encoder-decoder setups. Finally, I show that GraphWriter's transformer-style encoder is more effective than Graph Attention Networks on the knowledge-graph-to-text task.

	Title	Abstract	KG
Vocab	29K	77K	54K
Tokens	413K	5.8M	1.2M
Entities	-	-	518K
Avg Length	9.9	141.2	-
Avg #Vertices	-	-	12.42
Avg #Edges	-	-	4.43

Table 5.1: Data statistics of my AGENDA dataset. Averages are computed per instance.

5.1 The AGENDA Dataset

I consider the problem of generating a text from automatically extracted information (*knowledge*). IE systems can produce high quality knowledge for a variety of domains, synthesizing information from across sentence and even document boundaries. Generating coherent text from knowledge requires a model which considers global characteristics of the knowledge as well as local characteristics of each entity. This feature of the task motivates my use of graphs for representing knowledge, where neighborhoods localize important information and paths through the graph build connections between distant nodes through intermediate ones. An example knowledge graph can be seen in Figure 5.1.

I formulate my problem as follows: given the title of a scientific article and a knowledge graph constructed by an automatic information extraction system, the goal is to generate an abstract that a) is appropriate for the given title and b) expresses the content of the knowledge graph in natural language text. To evaluate how well a model accomplishes this goal, I introduce the Abstract GENeration DAtaset (AGENDA), a dataset of knowledge graphs paired with scientific abstracts. My dataset consists of 40k paper titles and abstracts from the Semantic Scholar Corpus taken from the proceedings of 12 top AI conferences [Ammar et al., 2018].

For each abstract, I create a knowledge graph in two steps. First, I apply the SciIE system of Luan et al. [2018], a state-of-the-art science-domain information extraction system. This system provides named entity recognition for scientific terms, with entity types Task, Method, Metric,

Material, or Other Scientific Term. The model also produces co-reference annotations as well as seven other relations that can obtain between entities (Compare, Used-for, Feature-of, Hyponym-of, Evaluate-for, and Conjunction).

I then form these annotations into knowledge graphs. I collapse co-referential entities into a single node associated with the longest mention (on the assumption that these will be the most informative). I then connect nodes to one another using the relation annotations, treating these as labeled edges in the graph. The result is a possibly unconnected graph representation of the SciIE annotations for a given abstract.

Statistics of the AGENDA dataset and comparison to related text generation datasets is available in Table 5.1. I split the AGENDA dataset into 38,720 training datapoints, with 1k each used for validation and test. I offer standardized data splits to facilitate comparison.

5.2 Model

Following most work on neural generation I adopt an encoder-decoder architecture, shown in Figure 5.2, which I call GraphWriter. The input to GraphWriter is a title and a knowledge graph which are encoded respectively with a bidirectional recurrent neural network and a novel Graph Transformer architecture (to be discussed in Section 5.2.1). At each decoder time step, I attend on encodings of the knowledge graph and document title using the decoder hidden state $\mathbf{h}_t \in \mathbb{R}^d$. The resulting vectors are used to select output w_t either from the decoder's vocabulary or by copying an entity from the knowledge graph. Details of my decoding process are described in Section 5.2.2. The model is trained end-to-end to minimize the negative log likelihood of the mixed copy and vocabulary probability distribution and the human authored text.

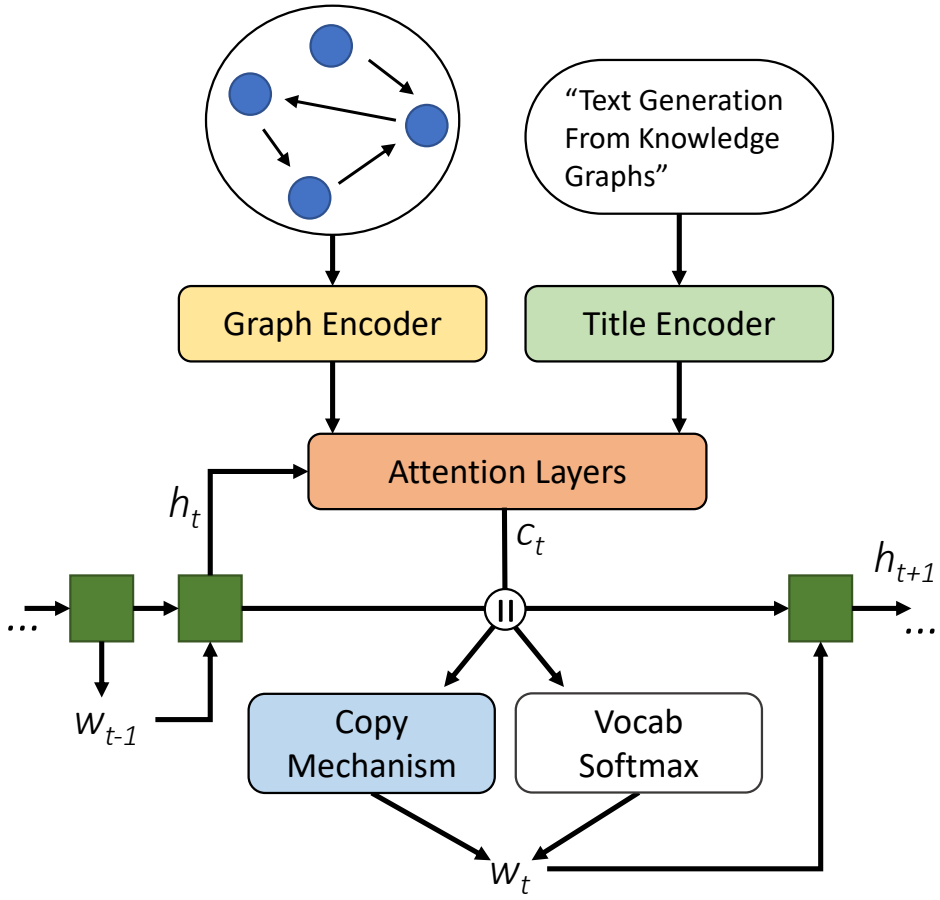

Figure 5.2: GraphWriter Model Overview

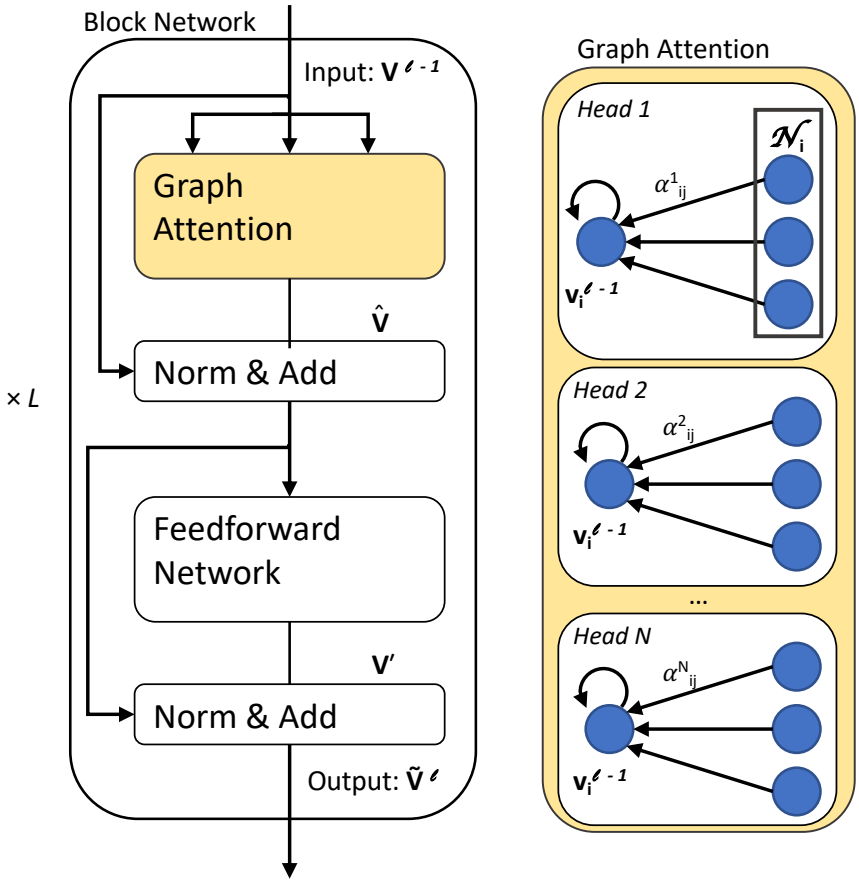

Figure 5.3: Graph Transformer

5.2.1 Encoding

The AGENDA dataset contains a knowledge graph for each datapoint, but the attention-based graph encoders we use here operate over unlabeled graphs. I convert each graph to an equivalent unlabeled bipartite graph following a similar procedure to Beck et al. [2018]. The details are described in Section 5.2.3 and sketched in Figure 5.4. The final result of these conversions is a connected, unlabeled graph $G = (V, E)$, where V is a list of entities, relations, and a global node and E is an adjacency matrix describing the edges.

Graph Transformer Encoder My model is most similar to the Graph Attention Network (GAT) of Veličković et al. [2018], which computes the hidden representations of each node in a graph by attending over its neighbors following a self-attention strategy. The use of self-attention in GAT addresses the shortcomings of prior methods based on graph convolutions [Defferrard et al., 2016; Kipf and Welling, 2017], but limits vertex updates to information from adjacent nodes. My model allows for a more global contextualization of each vertex through the use of a transformer-style architecture. The recently proposed Transformer [Vaswani et al., 2017] addresses the inherent sequential computation shortcoming of recurrent neural networks, enabling efficient and paralleled computation by invoking a self-attention mechanism for global context modeling. These models have shown promising results in a variety of text processing tasks [Radford et al., 2018].

My Graph Transformer encoder starts with self-attention of local neighborhoods of vertices; the key difference with GAT is that my model includes additional mechanisms for capturing global context. This additional modeling power allows the Graph Transformer to better articulate how a vertex should be updated given the content of its neighbors, as well as to learn global patterns of graph structure relevant to the model's objective.

Specifically, V is embedded in a dense continuous space by the embedding process described at the end of this section, resulting in matrix $\mathbf{V}^0 = [\mathbf{v}_i], \mathbf{v}_i \in \mathbb{R}^d$ which will serve as input to the graph transformer model shown in Figure 5.3. Each vertex representation $\mathbf{v}_i$ is contextualized by

attending over the other vertices to which v_i is connected in G. I use an N-headed self attention setup, where N independent attentions are calculated and concatenated before a residual connection is applied:

$$\hat{\mathbf{v}}_i = \mathbf{v}_i + \Big\|_{n=1}^{N} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^n \mathbf{W}_V^n \mathbf{v}_j \tag{5.1}$$

$$\alpha_{ij}^n = a^n(\mathbf{v}_i, \mathbf{v}_j) \tag{5.2}$$

Here, $\|$ denotes the concatenation of the N attention heads, $\mathcal{N}_i$ denotes the neighborhood of v_i in G, $\mathbf{W}_V^n \in \mathbb{R}^{d \times d}$, and where a^n are attention mechanisms parameterized per head. In this work, I use attention functions of the following form:

$$a(\mathbf{q}_i, \mathbf{k}_j) = \frac{\exp((\mathbf{W}_K \mathbf{k}_j)^\top \mathbf{W}_Q \mathbf{q}_i)}{\sum_{z \in \mathcal{N}_i} \exp((\mathbf{W}_K \mathbf{k}_z)^\top \mathbf{W}_Q \mathbf{q}_i)} \tag{5.3}$$

Each a learns independent transformations $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d}$ of $\mathbf{q}$ and $\mathbf{k}$ respectively, and the resulting product is normalized across all connected edges. To reduce the tendency of these dot products to impede gradient flow, I scale them by $\frac{1}{\sqrt{d}}$, following Vaswani et al. [2017].

The Graph Transformer then augments these multi-headed attention layers with *block* networks. Each block applies the following transformations:

$$\tilde{\mathbf{v}}_i = \text{LayerNorm}(\mathbf{v}'_i + \text{LayerNorm}(\hat{\mathbf{v}}_i)) \tag{5.4}$$

$$\mathbf{v}'_i = \text{FFN}(\text{LayerNorm}(\hat{\mathbf{v}}_i)) \tag{5.5}$$

Where $\text{FFN}(\mathbf{x})$ is a two layer feedforward network with a non-linear transformation f between layers i.e. $f(\mathbf{x}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$.

Stacking multiple blocks allows information to propagate through the graph. Blocks are stacked L times, with the output of layer $l-1$ taken as the input to layer l, so that $\mathbf{v}_i^l = \tilde{\mathbf{v}}_i^{l-1}$. The resulting

vertex encodings $\mathbf{V}^L = [\mathbf{v}_i^L]$ represent entities, relations, and the global node contextualized by their relationships in the graph structure. I refer to the resulting encodings as *graph contextualized vertex encodings*.

Embedding Vertices, Encoding Title As stated above, the vertices of my graph correspond to entities and relations from the SciIE annotations. Because each relation is represented as both a forward- and backward-looking vertex, I learn two embeddings per relation as well as an initial embedding for the global node. Entities correspond to scientific terms which are often multi-word expressions. To produce a single d-dimensional embedding per phrase, I use the last hidden state of a bidirectional RNN run over embeddings of each word in the entity phrase, i.e. $\text{BiRNN}(\mathbf{x}_1 \dots \mathbf{x}_m)$ for dense embeddings $\mathbf{x}$ and phrase length m. The output of my embedding step is a collection $\mathbf{V}^0$ of d-dimensional vectors representing each vertex in V.

The title input is also a short string, and so I encode it with another BiRNN to produce $\mathbf{T} = \text{BiRNN}(x'_1 \dots x'_m)$ for title word embedding $\mathbf{x}'$.

5.2.2 Decoding

I decode with attention-based decoder with a copy mechanism for copying input from the knowledge graph and title. At each decoding timestep t I use decoder hidden state $\mathbf{h}_t$ to compute context vectors $\mathbf{c}_g$ and $\mathbf{c}_s$ for the graph and title sequence respectively. $\mathbf{c}_g$ is computed using multi-headed attention contextualized by $\mathbf{h}_t$:

$$\mathbf{c}_g = \mathbf{h}_t + \Big|\Big|_{n=1}^{N} \sum_{j \in V} \alpha_j^n \mathbf{W}_G^n \mathbf{v^L}_j \tag{5.6}$$

$$\alpha_j = a(\mathbf{h}_t, \mathbf{v^L}_j) \tag{5.7}$$

for a as described in Equation 5.1 by attending over the graph contextualized encodings $\mathbf{V}^L$. $\mathbf{c}_s$ is computed similarly, attending over the title encoding $\mathbf{T}$. I then construct the final context vector by

concatenation, $\mathbf{c}_t = [\mathbf{c}_g \| \mathbf{c}_s]$. I use an input-feeding decoder [Luong et al., 2015] where both $\mathbf{h}_t$ and $\mathbf{c}_t$ are passed as input to the next RNN timestep.

I compute the probability p of copying from the input using $\mathbf{h}_t$ and

$$p = \sigma(\mathbf{W}_{copy}[\mathbf{h}_t \| \mathbf{c}_t] + b_{copy}) \tag{5.8}$$

The final next-token probability distribution is:

$$p * \alpha^{copy} + (1 - p) * \alpha^{vocab}, \tag{5.9}$$

Where the probability distribution α^{copy} over entities and input tokens is computed as $\alpha_j^{copy} = a([\mathbf{h}_t \| \mathbf{c}_t], \mathbf{x}_j)$ for $\mathbf{x}_j \in \mathbf{V} \| \mathbf{T}$. The remaining $1 - p$ probability is given to α^{vocab}, which is calculated by scaling $[\mathbf{h}_t \| \mathbf{c}_t]$ to the vocabulary size and taking a softmax.

5.2.3 Graph Preparation

Here I describe the details of the graph preparation step which converts each graph to an equivalent unlabeled connected bipartite graphs following a similar procedure to Beck et al. [2018]. In this process, each labeled edge is replaced with two vertices: one representing the forward direction of the relation and one representing the reverse. These new vertices are then connected to the entity vertices so that the directionality of the former edge is maintained. The result is an unlabeled directed graph where all vertices correspond to entities and relations in the SciIE annotations. To promote information flow between disconnected parts of the graph, I add a global vertex which connects all entity vertices. It will be used to initialize the decoder, analogously to the final encoder hidden state in a traditional sequence to sequence model.

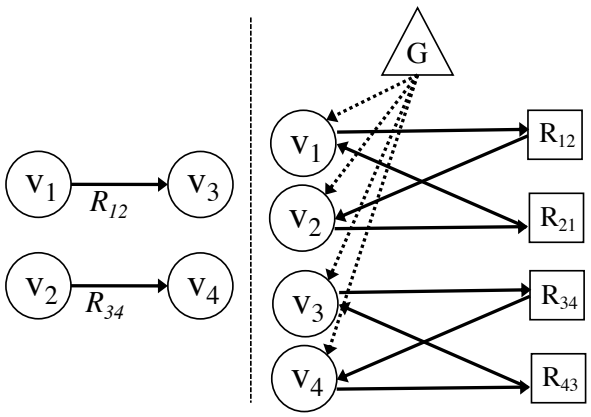

Figure 5.4: Converting disconnected labeled graph to connected unlabeled graph for use in attention-based encoder. v_i refer to vertices, R_{ij} to relations, and G is a global context node.

5.3 Experiments

Evaluation Metrics I evaluate using a combination of human and automatic evaluations. For **human** evaluation, participants were asked to compare abstracts generated by various models and those written by the authors of the scientific articles. I used Best-Worst Scaling (BWS; [Louviere and Woodworth, 1991; Louviere et al., 2015]), a less labor-intensive alternative to paired comparisons that has been shown to produce more reliable results than rating scales [Kiritchenko and Mohammad, 2016]. Participants were presented with two or three abstracts and asked to decide which one was better and which one was worse in order of grammar and fluency (is the abstract written in well-formed English?), coherence (does the abstract have an introduction, state the problem or task, describe a solution, and discuss evaluations or results?), and informativeness (does the abstract relate to the provided title and make use of appropriate scientific terms?). I provided examples of good and bad abstracts and explain how they succeed or fail to meet the defined criteria. An example of my solicitation materials can be found in the appendix to this work.

Because my dataset is scientific in nature, evaluations must be done by experts and I can only collect a limited number of these high quality datapoints.[2] The study was conducted by 15 experts (i.e. computer science students) who were familiar with the abstract writing task and the content

[2]Attempts to crowd source this evaluation failed.

of the abstracts they judged. To supplement this, I also provide **automatic** metrics. I use BLEU [Papineni et al., 2002], an n-gram overlap measure popular in text generation tasks as well as machine translation tasks, and METEOR [Denkowski and Lavie, 2014], a similar metric with paraphrase and language-specific considerations.

Comparisons I compare my GraphWriter against several strong baselines. In GAT, I reduce the block structure of the encoder to just graph attention with PReLU activations between layers. To determine the usefulness of including graph relations, I compare to a model which uses only entities and title (EntityWriter). Finally, I compare against a model which uses only title information, the gated rewriter model from the codebase published by Wang et al. [2018] (Rewriter).[3]

Implementation Details My models are trained end-to-end to minimize the negative joint log likelihood of the target text vocabulary and the copied entity indices. I use SGD optimization with momentum [Qian, 1999] and "warm restarts", a cyclical regiment that reduces the learning rate from 0.25 to 0.05 over the course of 5 epochs, then resets for the following epoch. Models are trained for 15 epochs with early stopping [Prechelt, 1998] based on the validation loss, with most models stopping between 8 and 13 epochs. I use single-layer LSTMs [Hochreiter and Schmidhuber, 1997] as recurrent networks. I use dropout [Srivastava et al., 2014] in self attention layers set to 0.3. Hidden states and embedding dimensions are fixed at 500 and attentions learn 500 dimensional projections. In Block layers, the feedforward network has an intermediate size of 2000, and I use a PReLU activation function [He et al., 2015]. GraphWriter and GAT use $L = 6$ layers. The number of attention heads is set to 4. In all models, for both inputs and output, I replace words occurring fewer than 5 times with $<unk>$ tokens.[4] In each abstract, I replace all mentions in a coreference chain in the abstract with the canonical mention used in the graph. I decode with beam search

[3]Due to the larger size and greater variety of my dataset and accompanying vocabularies compared to theirs, I was unable to train this model with the reported batch size of 240. I use batch size 24 instead, which is partially responsible for the lower performance.

[4]This threshold is typical in related literature, and was selected so as to result in an output vocabulary between 10k and 20k words

	BLEU	METEOR
GraphWriter	**14.3** $\pm$ 1.01	**18.8** $\pm$ 0.28
GAT	12.2 $\pm$ 0.44	17.2 $\pm$ 0.63
EntityWriter	10.38	16.53
Rewriter	1.05	8.38

Table 5.2: Automatic Evaluations of Generation Systems.

[Graves, 2012; Sutskever et al., 2014] with a beam size of 4. A post-processing step deletes repeated sentences and repeated coordinated clauses.

5.3.1 Results

A comparison of all systems in terms of automatic metrics is shown in Table 5.2. My GraphWriter model outperforms other methods. I see that models which leverage title, entities, and relations (GraphWriter and GAT) outperform models which use less information (EntityWriter and Rewriter).

I see that GraphWriter outperforms GAT across metrics, indicating that the global contextualization provided by GraphWriter improves generation. To verify the performance gap between GraphWriter and GAT, I report the average automatic metrics for 4 training runs of each model along with their variances. I see that the variance of the different models is non-overlapping, and in fact all training runs of GraphWriter outperformed all runs of GAT on these metrics.

Does Knowledge Help? To evaluate the value of knowledge in the generation task I compare my GraphWriter model to a model which does not generate from knowledge. I provide expert annotators with 50 randomly-selected paper titles from the test set and ask them for a single judgment according to the criteria described in Section 5.3. I pair each paper title with the generated abstracts produced by GraphWriter (a knowledge-informed modes), Rewriter (a knowledge-agnostic model), and the gold abstract (with canonicalized coreferential mentions, i.e. all mentions in a coreference chain are replaced with the single longest mention).

Results of this comparison can be seen in Table 5.3. I see that GraphWriter is selected as "Best"

	Best	Worst
Rewriter (No knowledge)	6	32
GraphWriter (Knowledge)	12	18
Human Authored	32	0

Table 5.3: Does knowledge improve generation? Human evaluations of best and worst abstract.

	Win	Lose	Tie
Structure	63%	17%	20%
Informativeness	43%	23%	33%
Grammar	63%	23%	13%
Overall	63%	17%	20%

Table 5.4: Human Judgments of GraphWriter and EntityWriter models.

more often than Rewriter, and is less often selected as "Worst", attesting to the value of including knowledge in the text generation process. I see that sometimes generated texts are preferred to human authored text, which is due in part to the disfluencies introduced by canonicalization of entity mentions.

To further understand the advantages of using knowledge graphs, I provide a more detailed comparison of the GraphWriter and EntityWriter models. I select 30 additional test datapoints and ask experts to provide per-criterion judgments of the outputs of the two systems. Since both models make use of extracted entities, I show this list along with the title for each datapoint, and modify the description of Informativeness to include "making use of the provided entities". Results of this evaluation are shown in Table 5.4. Here I see that including structured knowledge in the form of a graph improves abstract generation compared to generating from an unstructured collection of entities. The largest gains are made in terms of document structure and grammar, indicating that the structure of the input knowledge is being translated into the surface form.

Generating from Title The Rewriter model [Wang et al., 2018] considers the task of generating an abstract with only the paper's title as input. I compare against this model because it is among the first end-to-end systems to attempt to write scientific abstracts. However, the task setup used in

Title	Block and Group Regularized Sparse Modeling for Dictionary Learning
Knowledge	(dictionary learning, CONJUNCTION, sparse coding) ; (optimization problems, USED-FOR, dictionary learning) ; (optimization problems, USED-FOR, sparse coding)...
GraphWriter	sparse representations have recently been shown to be effective in many optimization problems. however, existing dictionary learning methods are limited in the number of dictionary blocks, which can be expensive to obtain. in this paper, we propose a novel approach to dictionary learning based on sparse coding ...
GAT	in this paper, we consider the problem of dictionary learning in well-known datasets. in particular, we consider the problem of dictionary learning, where the goal is to find a set of dictionary blocks that maximize the likelihood of a given set of dictionary blocks ...
EntityWriter	we propose a novel dictionary learning framework for reconstructed block/group sparse coding schemes. the dictionary learning framework is based on the descent, which is a block structure of the group structure ...
Rewriter	this paper presents a new approach to the k-means of the algorithm. the proposed approach is based on the basis of the stationarity algorithm. the algorithm is based on the fact that the number of bits is a constant of the base of the base of the input ...
Gold	This paper proposes a dictionary learning framework that combines the proposed block/group (BGSC) or reconstructed block/group (R-BGSC) sparse coding schemes with the novel Intra-block Coherence Suppression Dictionary Learning algorithm. An important and distinguishing feature of the proposed framework is that all dictionary blocks are trained simultaneously ...

Table 5.5: Example outputs of various systems versus Gold.

Wang et al. [2018] differs significantly from the task introduced in this thesis. In order to make a fair comparison, I construct a variant of my model which is only provided with a title as input. I develop a model that predicts entities from the title, and then uses my knowledge-aware model to generate the abstract. For this comparison I use the EntityWriter model with a collection of entities inferred from the title alone (InferEntityWriter).

To infer relevant entities, I learn to embed titles and entities extracted from the corresponding abstract in a shared dense vector space by minimizing their cosine distance. I use negative sampling to provide definition to this vector space.

At test time, I use the title embedding to infer the $K = 12$ closest entities to feed into the InferEntityWriter model. Results are shown in Table 5.6, which shows that InferEntityWriter achieves better results than Rewriter, indicating that the intermediate entity prediction step is helpful in abstract generation.

	BLEU	METEOR
Rewriter	1.05	8.38
InferEntityWriter	3.60	12.2

Table 5.6: Comparison of generation without knowledge and with Inferred Knowledge (InferEntityWriter)

5.3.2 Analysis

Table 5.5 shows examples of various system outputs for a particular test instance. I see that GraphWriter makes use of more entities from the input, arranged with more articulated textual context. It demonstrates less repetition than GAT. Both GraphWriter and GAT show much better coherence than EntityWriter, which copies entities from the input into unreasonable contexts. Rewriter, while fluent and grammatical, jumps from topic to topic, failing to relate as strongly to the input as the knowledge-aware models.

Errors To determine the shortcomings of my model, I calculate rough error statistics over the outputs of the GraphWriter on the test set. I notice that 40% of entities in the knowledge graphs do not appear in the generated text. Future work should address this coverage problem, perhaps through modifications to the inference procedure or a coverage loss modified to the specifics of this task. I find that 18% of all sentences generated by my model repeat sentences or clauses and are subjected to the post-processing pruning mentioned in Section 5.3. While this step is a simple solution to improve generated outputs, a more advanced solution is required as sometimes information is repeated in different syntactic configurations. Such syntactic variation is not accounted for by the simple post-processing step used here.

5.4 Conclusion

I have studied the problem of generating multi-sentence text from the output of automatic information extraction systems, and have shown that incorporating knowledge as graphs improves performance. I introduced GraphWriter, featuring a new attention model for graph encoding, and demonstrated its utility through human and automatic evaluation compared to strong baselines. Lastly, I provide a new resource for the generation community, the AGENDA dataset of abstracts and knowledge. Future work could address the problem of repetition and entity coverage in the generated texts.

Chapter 6

Conclusion

In this thesis, I have addressed specific problems in multi-sentence text understanding and generation with an eye toward providing as general solutions as possible. The solutions I proposed demonstrate to varying degrees each model's understanding of a small collection of concepts and their relationship to a text.

In Chapter 3, I described a model for automatically solving open-world math word problems using cross-sentence reasoning. Each math word problem is modeled as a semantically-enhanced equation tree using a novel recursive semantic structure called a *qset* which records semantic information about the quantities involved. With this formalism, the model generates a collection of possible semantically-enhanced equation trees using an integer linear programming approach. By learning to score local combinations of qsets and the likelihood of a tree based on learned features and violated ILP constraints, I select the maximum scoring tree to answer each problem.

In Chapter 4, I described the ThemeRewriter method for automatically customizing math word problems to meet thematic constraints. ThemeRewriter leverages the complex document structure of human authored text by editing in a globally coherent and syntactically informed way. I show that ThemeRewriter can be applied to the task of customizing math word problems to fit an arbitrary theme of student interest. ThemeRewriter identifies the most thematic words and characters from

a provided text and uses them to construct a collection of candidate rewrites of a word problem. Because syntactic and semantic relationships build document coherence and the mathematical meaning of the text, ThemeRewriter balances its thematic substitutions against the text's existing syntactic and semantic relationships. Automatic and human experiments show that ThemeRewriter produces the best thematic rewrite of the problem while preserving its mathematical intention.

In Chapter 5, I provided the GraphWriter method for generating multi-sentence texts from knowledge graphs using an innovative neural encoding technique. I show that knowledge graphs can serve as document plans when generating longer texts. GraphWriter is powered by a novel graph transforming encoder which extends the recent transformer model for text encoding to graph-structured inputs. GraphWriter learns to encode the input graph and output text in an end to end fashion. I also introduce a dataset of 40k knowledge graphs extracted from scientific abstracts for future researchers use. Human and automatic evaluation shows that GraphWriter is capable of using the relational knowledge of the input to improve generated text compared to models which cannot leverage this knowledge.

This work approached two primary research questions: one regarding the combining of local and global information; and the other on possible pragmatic formalisms. In particular, I sought methods for combining local and global information when understanding or generating multi-sentence texts. The techniques of Chapters 3 and 4 use mixture models to combine these sources of information. The GraphWriter model introduced in Chapter 5 uses an attention mechanism to combine the global graph encoding with the local decoder context. Both mixture models and attention mechanisms have proved to be successful at combining local and global information in these tasks.

I was also interested to know how we can represent the pragmatic meaning of a text. In this thesis I explored three options: equation trees, which can aid in understanding existing math word problems; syntactic scaffolding, which can allow us to generate novel thematic math problems; and knowledge graphs, from which we can generate longer scientific abstracts. I believe that knowledge graphs are sufficiently general to be used in a variety of text understanding and generation scenarios.

Future Work There are several ways to build on the work outlined here. The graph transforming encoder introduced in Chapter 5 could be applied to other graph encoding tasks, such as knowledge base completion [Toutanova et al., 2015]. More ambitious would be producing pretrained embeddings of named entities from a large knowledge graph. The graph transformer model could be trained on a large graph such as Freebase [Bollacker et al., 2008] to predict masked neighbor nodes, similar to the training objective of the popular BERT embeddings [Devlin et al., 2018].

Another avenue for future work would involve theme-rewriting (or rewriting documents under other constraints) at a deeper semantic level. Instead of operating on surface text directly, thematic edits could be made to a knowledge graph corresponding to a word problem text. Surface text could then be recovered with the GraphWriter system of Chapter 5. A knowledge graph for general text can be produced by a semantic role labeling system such as He et al. [2017] with coreference annotations. For a math word problem, one can include mathematical relations between quantities as provided by an equation parser (Chapter 3). This proposed model has the potential to be more robust than the model described in Chapter 4 as it will be able to tailor the syntax of the problem to the content words provided by the editor. However, significantly more data will be required to learn the syntactic patterns that are present in a human authored text and absent in a knowledge graph. Pretraining should be considered to help fill this gap.

Bibliography

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Lu Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the Literature Graph in Semantic Scholar. In *NAACL*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *TACL*, 1(1):49–62.

Leandro Motta Barros and Soraia Raupp Musse. 2007. Planning algorithms for interactive storytelling. *Computers in Entertainment (CIE)*, 5(1):4.

Lora Bartlett. 2004. Expanding teacher work roles: a resource for retention or a recipe for overwork? *Journal of Education Policy*, 19(5):565–582.

Regina Barzilay and Mirella Lapata. 2005. Collective Content Selection for Concept-to-Text Generation. In *EMNLP*, pages 331–338. Association for Computational Linguistics.

Daniel Edward Robert Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *ACL*.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. 2015. Layers of Interpretation: On Grammar and Compositionality. In *Proceedings of the 11th*

International Conference on Computational Semantics, pages 239–249, London, UK. Association for Computational Linguistics.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling Biological Processes for Reading Comprehension. In *EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD*, pages 1247–1250, New York, NY, USA. ACM.

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label Ranking under Ambiguous Supervision for Learning Semantic Correspondences. In *ICML*, pages 103–110.

S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement Learning for Mapping Instructions to Actions. In *ACL/AFNLP*, pages 82–90.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and Their Participants. In *ACL/AFNLP*, pages 602–610. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *EMNLP*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

David Chen and William B. Dolan. 2011. Collecting Highly Parallel Data for Paraphrase Evaluation. In *ACL*.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a Multilingual Sportscaster: Using Perceptual Context to Learn Language. *JAIR*, 37:397–435.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *ICML*, pages 128–135.

Michael Collins. 2005. Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, 31(1):25–70.

The Racial Slur Database. 2016. The Racial Slur Database. Available at `http://www.rsdb.org/`.

Judy Davis-Dorsey, Steven M Ross, and Gary R Morrison. 1991. The role of rewording and context personalization in the solving of mathematical word problems. *Journal of Educational Psychology*, 83(1):61.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Marie-Catherine De Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*.

Michael J. Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Workshop on Statistical Machine Translation*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every Picture Tells a Story: Generating Sentences from Images. In *ECCV*.

Yansong Feng and Mirella Lapata. 2010. How Many Words is a Picture Worth? Automatic Caption Generation for News Images. In *ACL*, pages 1239–1249.

K. Filippova and M. Strube. 2008. Dependency Tree Based Sentence Compression. In *Proceedings of the Fifth International Natural Language Generation Conference (INLG))*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *ACL*, pages 363–370.

Dan Flickinger. 2000. On building a more effcient grammar by exploiting types. *Natural Language Engineering*, 6(01):15–28.

Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. *Language from a cognitive perspective: Grammar, usage, and processing*, pages 31–50.

J. Ganitkevitch, C. Callison-Burch, C. Napoles, and B. Van Durme. 2011. Learning Sentential Paraphrases from Bilingual Parallel Corpora for Text-to-Text Generation. In *EMNLP*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *NAACL*, pages 758–764.

Ruifang Ge and Raymond J. Mooney. 2005. A Statistical Semantic Parser that Integrates Syntax and Semantics. In *Conference on Computational Natural Language Learning*, pages 9–16.

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative Reranking for Semantic Parsing. In *ACL*.

Yoav Goldberg and Jon Orwant. 2013. A Dataset of Syntactic-Ngrams over Time from a Very Large Corpus of English Books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 1, pages 241–247.

D. Goldwasser and D. Roth. 2011. Learning from Natural Instructions. In *IJCAI*.

Philip John Gorinski and Mirella Lapata. 2015. Movie Script Summarization as Graph-based Scene Extraction. In *ACL*.

Mark Granroth-Wilding and Stephen Clark. 2016. What Happens Next? Event Prediction Using a Compositional Neural Network Model. In *AAAI*, Phoenix, Arizona.

Alex Graves. 2012. Sequence Transduction with Recurrent Neural Networks. *arXiv preprint arXiv:1211.3711*.

H. P. Grice. 1968. Utterers meaning, sentence-meaning, and word-meaning. *Foundations of Language*, 4(3):225 242.

Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. 2011. Reasoning about RoboCup Soccer Narratives. In *UAI*, pages 291–300.

Hannaneh Hajishirzi, Mohammad Rastegari, Ali Farhadi, and Jessica Hodgins. 2012. Semantic Understanding of Professional Soccer Commentaries. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Janis M Hart. 1996. The Effect of Personalized Word Problems. *Teaching Children Mathematics*, 2(8):504–505.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of ACL*.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning Knowledge Graphs for Question Answering through Conversational Dialog. In *NAACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to Solve Arithmetic Word Problems with Verb Categorization. In *EMNLP*, pages 523–533.

IBM ILOG. 2014. IBM ILOG CPLEX Optimization Studio 12.6.

David Kauchak. 2013. Improving Text Simplification Language Modeling Using Unsimplified Text Data. In *ACL*.

Joohyun Kim and Raymond J Mooney. 2010. Generative Alignment and Semantic Parsing for Learning from Ambiguous Supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Svetlana Kiritchenko and Saif Mohammad. 2016. Capturing Reliable Fine-Grained Sentiment Associations by Crowdsourcing and Best-Worst Scaling. In *NAACL-HLT*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation From Knowledge Graphs with Graph Transformers. In *NAACL-HLT*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi. 2014. Multi-Resolution Language Grounding with Weak Supervision. In *EMNLP*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Ang. 2015. Parsing Algebraic Word Problems into Equations. *TACL*, 3.

Rik Koncel-Kedziorski, Ioannis Konstas, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2016a. A Theme-Rewriting Approach for Generating Algebra Word Problems. In *EMNLP*, pages 1617–1628.

Rik Koncel-Kedziorski, Subhro Roy, Aida Aimini, Nate Kushman, and Hannaneh Hajishirzi. 2016b. MAWPS: A Math Word Problem Repository. In *NAACL-HLT*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke S. Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *ACL*.

Ioannis Konstas and Mirella Lapata. 2013. Inducing Document Plans for Concept-to-Text Generation. In *EMNLP*, pages 1503–1514.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L Berg. 2011. Baby Talk: Understanding and Generating Image Descriptions. In *CVPR*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to Automatically Solve Algebra Word Problems. In *ACL*, pages 271–281.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing Probabilistic CCG Grammars from Logical Form with Higher-Order Unification. In *EMNLP*, pages 1223–1233.

Benoit Lavoie and Owen Rambow. 1997. A Fast and Portable Realizer for Text Generation Systems. In *Applied Natural Language Processing*, pages 265–268.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *EMNLP*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *Proceedings of EMNLP*.

Friedrich Wilhelm Levi. 1942. *Finite geometrical systems: six public lectures delivered in February, 1940, at the University of Calcutta*. The University of Calcutta.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *ACL*, pages 302–308.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark O. Riedl. 2013. Story Generation with Crowdsourced Plot Graphs. In *AAAI*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A Diversity-Promoting Objective Function for Neural Conversation Models. *arXiv preprint arXiv:1510.03055*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning Semantic Correspondences with Less Supervision. In *ACL/AFNLP*, pages 91–99.

Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. In *ICML*, volume 98, pages 296–304.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *NAACL*.

Jordan J Louviere, Terry N Flynn, and Anthony Alfred John Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge University Press.

Jordan J Louviere and George G Woodworth. 1991. Best-Worst Scaling: A Model for the Largest Difference Judgments. *University of Alberta: Working Paper*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In *EMNLP*, pages 3219–3232.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL: System Demonstrations*, pages 55–60.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep Graph Convolutional Encoders for Structured Data to Text Generation. *INLG*.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2012. Learning to Parse Natural Language Commands to a Robot Control System. In *ISER*.

Neil McIntyre and Mirella Lapata. 2009. Learning to Tell Tales: A Data-driven Approach to Story Generation. In *ACL/AFNLP*, pages 217–225.

Neil McIntyre and Mirella Lapata. 2010. Plot Induction and Evolutionary Search for Story Generation. In *ACL*, pages 1562–1572.

FrontGate Media. 2016. Terms to block.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *NAACL-HLT*, pages 720–730.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

George A Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Arindam Mitra and Chitta Baral. 2015. Learning to Automatically Solve Logic Grid Puzzles. In *EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*.

Sarah Petersen and Mari Ostendorf. 2007. Text Simplification for Language Learners: A Corpus Analysis. In *Speech and Language Technology in Education Workshop*.

Karl Pichotta and Raymond J. Mooney. 2016. Learning Statistical Scripts with LSTMRecurrent Neural Networks. In *AAAI*, Phoenix, Arizona.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press and CSLI Publications, Chicago, IL and Stanford, CA.

Oleksandr Polozov, Eleanor O'Rourke, Adam M Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popović. 2015. Personalized Mathematical Word Problem Generation. In *IJCAI*.

Lutz Prechelt. 1998. Early Stopping - but when? In *Neural Networks: Tricks of the Trade*, pages 55–69, London, UK, UK. Springer-Verlag.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-Text Generation with Content Selection and Planning. In *AAAI*.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 12 1:145–151.

Willard Van Orman Quine. 1969. *Word and object*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. Accessed at https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ Questions for Machine Comprehension of Text. *EMNLP*.

KA Renninger, L Ewen, and AK Lasher. 2002. Individual interest as context in expository text and mathematical word problems. *Learning and Instruction*, 12(4):467–490.

Dan Roth and Wen-tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *CoNLL*, pages 1–8.

Subhro Roy and Dan Roth. 2015. Solving General Arithmetic Word Problems. In *EMNLP*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. *EMNLP*.

Roger C Schank and Robert P Abelson. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Hillsdale, NJ: Lawrence Erlbaum.

Alexander Schrijver. 1998. *Theory of linear and integer programming*. John Wiley & Sons.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram Understanding in Geometry Questions. In *AAAI*.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving Geometry Problems: Combining Text and Diagram Interpretation. In *EMNLP*.

SequencePublishing. 2016. https://www.sequencepublishing.com/1/academic/academic.html#function.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically Solving Number Word Problems by Semantic Parsing and Reasoning. In *EMNLP*.

Advaith Siddharthan. 2004. Syntactic Simplification and Text Cohesion. *Research on Language and Computation*, 4(1):77–109.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A Graph-to-Sequence Model for AMR-to-Text Generation. In *ACL*.

Springfield. 2016. Adventure Time with Finn & Jake Episode Scripts. Available at `http://www.springfieldspringfield.co.uk/episode_scripts.php?tv-show=adventure-time`.

Vivek Srikumar and Dan Roth. 2011. A Joint Model for Extended Semantic Role Labeling. In *EMNLP*, Edinburgh, Scotland.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Ladder Survivors. 2013. Key words for math problems.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*, pages 3104–3112.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *EMNLP*, pages 1499–1509.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *NAACL-HLT*, pages 173–180. Association for Computational Linguistics.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond SumBasic: Task-focused Summarization with Sentence Simplification and Lexical Expansion. *Information Processing and Management*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NeurIPS*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

David Vickrey and Daphne Koller. 2008. Sentence Simplification for Semantic Role Labeling. In *ACL*, pages 344–352.

Adam Vogel and Daniel Jurafsky. 2010. Learning to Follow Navigational Directions. In *ACL*, pages 806–814.

Qingyun Wang, Zhihao Zhou, Lifu Huang, Spencer Whitehead, Boliang Zhang, Heng Ji, and Kevin Knight. 2018. Paper Abstract Writing through Editing Mechanism. In *Proceedings of NAACL-HLT*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing Datasets for Multi-Hop Reading Comprehension Across Documents. *TACL*, 6:287–302.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in Data-to-document Generation. In *EMNLP*.

Kristian Woodsend and Mirella Lapata. 2011a. Learning to Simplify Sentences with Quasi-SynchronousGrammar and Integer Programming. In *EMNLP*.

Kristian Woodsend and Mirella Lapata. 2011b. Wikisimple: Automatic simplification of wikipedia articles. In *Proceedings of the Association for Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*, pages 927–932, San Francisco, CA.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence Simplification by Monolingual Machine Translation. In *ACL*, pages 1015–1024.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing Statistical Machine Translation for Text Simplification. *TACL*.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the Sake of Simplicity: Unsupervised Extraction of Lexical Simplifications from Wikipedia. In *NAACL-HLT*.

Mark Yatskar, Lucy Vanderwende, and Luke Zettlemoyer. 2014. See No Evil, Say No Evil: Description Generation from Densely Labeled Images. *Lexical and Computational Semantics (* SEM 2014)*, page 110.

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *AAAI*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *UAI*, pages 658–666.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to Solve Algebra Word Problems Using Quadratic Programming. In *EMNLP*.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In *COLING*.