Selecting and Generating Computational Meaning Representations for Short Texts

by

Catherine Finegan-Dollak

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2018

Doctoral Committee:

Professor Dragomir Radev, Co-Chair, Yale University
Assistant Professor Walter Lasecki, Co-Chair
Associate Professor Ezra Keshet
Professor Rada Mihalcea

Catherine Finegan-Dollak

cfdollak@umich.edu

ORCID iD: 0000-0003-4316-565X

DEDICATION

To Jordan, who believes I could have written a dissertation without his help and support, but is mistaken.

ACKNOWLEDGMENTS

This dissertation would never have been possible without the help, advice, feedback, and support of many people.

I would like to thank my advisor, Dragomir Radev, who took a risk and accepted a PhD student who did not have the background many professors would have considered necessary. I have benefited enormously from his breadth of knowledge of natural language processing and his generosity with his time. I would not have become the researcher I am today without him.

I am grateful also to Walter Lasecki, Rada Mihalcea, and Ezra Keshet for serving as my thesis committee.

During my time at the University of Michigan, I have had the opportunity to work with a number of wonderful collaborators. Thank you to all of the members of the CLAIR research group, as well as to all of the members of the Project Sapphire team, especially Jonathan Kummerfeld.

Financial support for this work was provided in part by the Richard H. Orenstein Fellowship and by IBM; I am grateful it.

Very special thanks to the friends and family who have supported me, listened to me rant, and provided baked goods when needed. Thanks to my parents for doing their best to understand why I would want to go back to school yet again. Thanks to my grandmother, who prayed a *lot* of rosaries for me. Thanks to my brother and my nephew for study breaks over Skype. Thanks to Ada, the only dog I know of who recognizes the word "dissertate." And above all, thank you, thank you, thank you to my wonderful husband Jordan, who has helped in more ways than I can count.

CONTENTS

Dedication . . . . . **ii**

Acknowledgments . . . . . **iii**

List of Figures . . . . . **x**

List of Tables . . . . . **xii**

Abstract . . . . . **xiii**

Chapter

1 Introduction **1**

1.1 How to Represent Meaning 2

1.1.1 Overview of Existing MRs 2

1.1.2 Composing to Sentences 7

1.1.3 Grounding 10

1.1.4 Natural Language as Its Own MR 12

1.1.5 Applications of Meaning Representations 13

1.1.6 Semantic Relationships between Sentences 18

1.2 Transforming Text to MRs 19

1.2.1 Semantic Parsing 20

1.2.2 Deep Learning Approaches 22

1.3 The Present Work 25

1.3.1 What Meaning Representations Should We Use? 25
1.3.2 How Can We Transform Text to a Meaning Representation? . . . 27

Part I Selecting a Meaning Representation 31

2 Sentence Compression, Simplification, and Disaggregation for Sophisticated Texts 32

2.1 Related Work 35
2.1.1 Summarization 35
2.1.2 Simplification, Compression, and Disaggregation 36
2.2 Problem Definition and Algorithms 40
2.2.1 Task Definition 40
2.2.2 Existing Systems 43
2.2.3 Disaggregation System 46
2.2.4 Summarization System 50
2.3 Experimental Evaluation 51
2.3.1 Evaluation Methodology 51
2.3.2 Results 54
2.4 Discussion 57
2.4.1 Analysis of Results on Quality Questions 57
2.4.2 Analysis of Results on Comprehension Questions 61
2.4.3 Qualitative Analysis of Shortened Sentences 64
2.4.4 Strategies for Improvement 66
2.5 Conclusion 69

3 Explorations of Paraphrase Detection 70

3.1 Related Work 71
3.1.1 Paraphrase Methods without Deep Learning 71

3.1.2 Deep Learning . . . 74
3.2 Methods . . . 76
3.2.1 Traditional LSTM . . . 76
3.2.2 Tree LSTMs . . . 77
3.2.3 Siamese Architecture . . . 79
3.2.4 Pretraining . . . 79
3.3 Experiments . . . 81
3.3.1 Baselines . . . 82
3.4 Results . . . 83
3.5 Potential Future Work . . . 84
3.5.1 Label-Sensitive Tree LSTM . . . 84
3.5.2 Additional Training Data . . . 84
4 Effects of Text Corpus Properties on Short Text Clustering Performance . . . 86
4.1 Related Work . . . 88
4.2 Datasets . . . 89
4.3 Method . . . 92
4.3.1 Similarity Metrics . . . 92
4.3.2 Clustering Methods . . . 95
4.3.3 Evaluation Methods . . . 96
4.4 Vocabulary Width . . . 97
4.4.1 Descriptive Statistics for Vocabulary Width . . . 97
4.4.2 Experiments . . . 98
4.4.3 Results and Discussion . . . 98
4.5 Tightness of Clusters . . . 100
4.5.1 Descriptive Statistics for Tightness . . . 100
4.5.2 Results and Discussion . . . 102
4.6 Conclusions and Future Work . . . 104

Part II From Text to Meaning Representation 106

5 Improving Text-to-SQL Evaluation Methodology . . . 107

5.1 Introduction . . . 107

5.2 Related Work . . . 108

5.3 Data . . . 110

5.3.1 SQL Canonicalization . . . 112

5.3.2 Variable Annotation . . . 112

5.3.3 Query Deduplication . . . 113

5.4 Dataset Characteristics Show Evaluating on Multiple Datasets Is Necessary 113

5.4.1 Measures . . . 114

5.4.2 Analysis . . . 115

5.5 Current Data Splits Only Partially Probe Generalizability . . . 117

5.5.1 Systems . . . 118

5.5.2 Oracle Entity Condition . . . 120

5.5.3 Results and Analysis . . . 121

5.5.4 Logic Variants . . . 124

5.6 Conclusion . . . 125

6 Schema-Aware Text-to-SQL . . . 126

6.1 Introduction . . . 126

6.2 Related Work . . . 128

6.3 Methods . . . 130

6.3.1 Schema Representations . . . 131

6.3.2 Seq2Seq Model . . . 132

6.3.3 Attention-Based Copying . . . 134

6.4 Datasets . . . 136

6.5 Experiments . . . 137

6.6 Results and Analysis . . . 138
6.6.1 Schema Attention and Copying . . . 138
6.6.2 Schema Maps . . . 140
6.7 Conclusions and Future Work . . . 140
7 Text-to-SQL in Dialog Context . . . 142
7.1 Introduction . . . 142
7.2 Dataset . . . 143
7.2.1 Preliminary Annotation . . . 144
7.2.2 Subsequent Annotation of "Exact" Utterances . . . 145
7.3 Pilot Experiments . . . 146
7.4 Results . . . 147
7.5 Future Work . . . 148
7.6 Conclusion . . . 149
8 Conclusion . . . 150
References . . . 155

LIST OF FIGURES

1.1 An example recurrent neural network for part-of-speech tagging. 23

1.2 An example seq2seq model for machine translation 24

2.1 Distributions of sentence lengths in three corpora 42

2.2 A rule from the simplification system 45

2.3 Dependency parse before and after simplification 45

2.4 Mean comprehension scores achieved by the different systems 55

2.5 DUC quality question scores 56

3.1 Parse trees of different sentences containing identical words 80

3.2 Parse trees with the same child-sum LSTMs 84

4.1 Image from PAS dataset 90

4.2 Image from New Yorker cartoon captioning contest 91

4.3 Plots of tfidf and word2vec representations of clusters from CLUE 101

4.4 Plots of tfidf and word2vec representations of clusters from PAS 101

4.5 All similarity metrics and all clustering methods for the four datasets. 103

5.1 Example of question-based and query-based dataset splits 108

5.2 SQL pattern and example queries 114

5.3 Template-based baseline system 119

6.1 Two schemas for a database that can answer the same question 126

6.2 Example schema embedding 131

6.3 Example schema map embedding 132

LIST OF TABLES

1.1 A simple CCG lexicon. 21

2.1 Comparison of long sentences in three corpora 41
2.2 Differences in output of the disaggregation system and the simplifier 50
2.3 Comprehension question examples 53
2.4 Word counts for documents and summaries 54
2.5 Mean comprehension scores for each condition 55
2.6 Correlation coefficients between comprehension scores and quality scores . . . 57
2.7 Samples of summaries output by the experimental systems 59

3.1 Results on the MSRP corpus. 83

4.1 Vocabulary properties of each dataset 97
4.2 Comparison of all similarity metrics on PAS and CLUE datasets, clustered using k-means and MCL 99
4.3 Modularity for all datasets 102

5.1 Manually identified duplicate queries 113
5.2 Descriptive statistics for text-to-SQL dataset diversity 115
5.3 Descriptive statistics for text-to-SQL dataset complexity 116
5.4 Accuracy of neural text-to-SQL systems on English question splits and SQL query splits 120
5.5 Types of errors by the attention-based copying model 121

6.1 Text-to-SQL dataset sizes 136
6.2 Accuracy of seq2seq models with and without schema attention and copying. . 138
6.3 Overlap in incorrect questions between experimental systems and baseline . . . 139
6.4 Accuracy with and without schema map 140

7.1 Labels used in the second-level review 145
7.2 Results of pilot experiments on a new dialog-to-SQL task 147

ABSTRACT

Language conveys meaning, so natural language processing (NLP) requires representations of meaning. This work addresses two broad questions: (1) What meaning representation should we use? and (2) How can we transform text to our chosen meaning representation? In the first part, we explore different meaning representations (MRs) of short texts, ranging from surface forms to deep-learning-based models. We show the advantages and disadvantages of a variety of MRs for summarization, paraphrase detection, and clustering. In the second part, we use SQL as a running example for an in-depth look at how we can parse text into our chosen MR. We examine the text-to-SQL problem from three perspectives—methodology, systems, and applications—and show how each contributes to a fuller understanding of the task.

CHAPTER 1

Introduction

Language is used to convey meaning, so of course many natural language processing (NLP) applications depend on the meaning of text. For example, summarization can be thought of as communicating the most important parts of a text's meaning. Question answering involves identifying a particular relationship between the meanings of two texts: one answers the other. Text generation requires starting from some desired meaning and creating a grammatically correct expression of that meaning in text. And machine translation can be viewed as the task of expressing the meaning of a text from one language in another language. Thus, the question of how to represent the meaning of text is an essential, underlying part of most of NLP. This thesis examines two aspects of that question.

The first is *choosing meaning representations* (MRs): What are our options? Should we simply use the text as its own representation? Should we rely on traditional bag-of-words-based representations? Should we parse sentences into lambda calculus or other structured MRs? Or should we rely on neural networks to generate some powerful but uninterpretable MR? And to what extent is our choice of MR determined by the NLP task?

The second is once we have selected a representation, how can we *parse text into our chosen MR*? For some MRs, such as a word-count vector, this is almost trivial. However, for more structured representations, it is anything but; semantic parsing has been an area of research for decades and remains an open problem. Part of the difficulty is that natural language can be ambiguous, so that the same literal string can mean different things in

different contexts: “She was admitted to the bar” might mean the bouncer allowed her into the drinking establishment or the state regulatory board licensed her as an attorney. Another challenge is the opposite: the same meaning may be expressed by different literal strings, called paraphrases.

As the related work will show, both the choice of MR and the method of generating it are extremely broad questions, and answering them in a single thesis is not possible. We therefore narrow the scope by limiting our exploration to short texts—sentences, fragments, and dialog snippets—leaving the wider world of lexical, paragraph-level, and document-level semantics aside. In addition, we focus our discussion of how to parse text on a single MR—in this case, SQL.

In this introduction, we begin with background information on how to represent meaning. This includes an overview of MRs that the NLP community has explored, followed by description of related work looking at the usefulness of MRs for a variety of tasks. We then review related research into how to transform text into the desired MR. Finally, we will describe how each of the subsequent chapters addresses the overarching questions: How should we represent meaning? And how can we parse text to our chosen meaning representation?

1.1 How to Represent Meaning

1.1.1 Overview of Existing MRs

Decades of NLP work have, implicitly or explicitly, required a way to represent the meaning of text. In this subsection, we briefly describe the most common MRs.

1.1.1.1 Starting from Words

Language is made up of words, and words typically contribute to a text's meaning. Thus, an obvious way to represent the meaning of a text is to start from words.

Word Counts and TF-IDF An early, and still frequently used, approach is simply to ignore syntax entirely and count the frequency of tokens[1] in a text. The text can then be represented as a word-count vector whose length is equal to the number of tokens in the vocabulary (often with an "UNK" token to represent any out-of-vocabulary tokens). The i-th value in the vector is the number of times the i-th token from the vocabulary appears in the text.

One problem with this is that certain words are very common and thus not very informative about the meaning of a particular sentence. For example, the word "the" is the most common word in English, appearing about twice as often as even the second most common word, yet knowing that the word "the" appears in a sentence tells us little about the sentence's meaning.

An alternative to word counts that takes this into account is *term-frequency inverse-document-frequency*, or tf-idf. As the name implies, this measure has two components. *Term frequency* is the same as the word counts above: how many times does each term in the vocabulary appear in the text we wish to represent? *Document frequency* refers to how often each term appears in the corpus as a whole. For words like "the," document frequency is a large number. Tf-idf is the term frequency times the inverse of the document frequency. Thus, while the word "the" may appear many times in a given text, it appears so often in most corpora that the tf-idf vector for the text will have a small value in the "the" slot of the vector.

Both word count vectors and tf-idf vectors have several weaknesses as MRs. For one thing, they are extremely sparse—that is, most of the values in the vector are zeros. The

[1] For simplicity, we focus on tokens, but this paragraph applies as well to bigrams, trigrams, and potentially longer n-grams.

number of words in a sentence is typically on the order of 10^1 or, for long sentences, 10^2, whereas a vocabulary size on the order of 10^4 is common. Another problem is that they represent sentences as a bag of words, which is to say they ignore syntax. "The dog chased the cat" and "The cat chased the dog" have identical word count and tf-idf vectors. And a third issue is that they ignore what words actually mean—*e.g.*, that dogs and cats are animals and that chasing is a type of movement.

Lexical semantics Lexical semantics moves beyond counting words and considers what individual words mean. On its surface, this seems like it should be straightforward: simply have a dictionary that lists the definition of each token in our vocabulary.

Yet this approach immediately encounters the problem of *polysemous* words, where the same token can have more than one meaning. For instance, in the sentence, "After visiting four *bars* in one night, John didn't remember much when he woke up behind *bars*," the same string of letters refers to both drinking establishments and metal barriers to keep prisoners in jail. Moreover, the second instance of *bars* may or may not be literal; "behind bars" may be used idiomatically to mean "in jail," even if the jail in question uses solid metal doors instead of bars to contain its prisoners. NLP work in *word sense disambiguation* and *metaphor detection* seek to address these issues.

A dictionary also has the problem of out-of-vocabulary (OOV) words. These may include infrequently used terms or neologisms—new words that did not exist when the dictionary was compiled. The Oxford English Dictionary publishes four updates each year to keep up with new words and new usages of old ones; their January 2018 update included over 1000 words.[2] In some cases, *morphology*—that is, the form of the word—can help. New words may be coined by combining old ones—for instance, one of the OED's new entries this year is *hangry*, which combines "hungry" and "angry." Affixes (in English, usually prefixes and suffixes) change the meanings of their roots in predictable ways; *undo*, *unpack*, and *unhook* bear similar relationships to *do*, *pack*, and *hook*, respectively. And

[2] http://public.oed.com/the-oed-today/recent-updates-to-the-oed/january-2018-update/

some methods of addressing OOV proper names are described in Subsection 1.1.3. Nevertheless, OOV words do remain a difficulty for dictionary-based representations of meaning.

Despite these challenges, dictionary-like databases of words have had an important role in NLP. In particular, WordNet (Fellbaum, 1998) is a widely used resource. It collects nouns, verbs, adjectives, and adverbs. Each word belongs to at least one *synset*, or grouping of words with the same or nearly the same meaning. Polysemous words appear in more than one synset, reflecting their multiple meanings.

Synsets are connected in a primarily hierarchical network structure. The majority of edges in the network represent hypernym/hyponym relations; the hypernym synset expresses a broader concept, and the hyponym synset expresses a specific form of that concept. For example, a synset containing "dog" is a direct hyponym of a synset containing "canine," which in turn is a direct hyponym of "carnivore." That synset for "carnivore" is hypernym not only to "canine," but also to "feline," which in turn is a hypernym to a synset containing the word "cat."

This network structure enables some measurement of the similarity between words. One straightforward method is to count the number of steps in the shortest path between two synsets; "dog" and "cat" are four steps apart. Other measures take into account other features of the nodes' location in the network, such as their depth (Leacock-Chodorow Similarity) or the Least Common Subsumer (LCS)—the closest ancestor that the two share (Wu-Palmer Similarity, Resnik Similarity).

For a (primarily) hand-curated resource, WordNet is enormous. Version 3.0 contains nearly 150,000 unique strings in over 115,000 synsets, creating a total of over 200,000 word-sense pairs.[3] It reflects decades of human effort. Alternative approaches that require less effort and can thus afford to capture a longer tail of word usages are obviously desirable; distributional semantics is one answer to this.

[3] http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html

Distributional Semantics Distributional semantics seeks to represent the meanings of words based on how they are used. These techniques look at large corpora, and, based on the contexts in which each word tends to appear, learn a large (typically 50-1000-dimensional) vector representation of each word. These vectors have some interesting and useful properties. Famously, if one starts with the vector for "king," subtracts the vector for "man," and adds the vector for "woman," the resulting vector is closer to the vector for "queen" that to any other word vector (Mikolov et al., 2013b). Likewise, the difference between the vectors for "Berlin" and "Germany" is quite similar to the difference between the vectors for "Paris" and "France" (Mikolov et al., 2013a). Although distributional semantics has been around for decades (Schütze, 1998), it has seen a rapid rise in popularity in recent years, in part thanks to the work of Mikolov, the availability of pre-trained word vectors (Mikolov et al., 2013a; Pennington et al., 2014), and the ability for word vectors to be used with deep neural networks.

Composing word vectors remains a significant challenge, though. Adding word vectors might seem like a reasonable place to start. Yet sentences and longer texts are often more than the sum of their parts. That is, summing the vectors treats sentences as bags of words without regard to syntax, just as word-count and tf-idf vectors do. Likewise, idioms and multiword expressions may mean something very different from what the sum of their vectors implies: one cannot simply add the vector for "cutting" to the vector for "corners" and hope to have an accurate representation of the meaning of "cutting corners." For some applications, summed embeddings provide good-enough performance with low overhead (Ritter et al., 2015); see also *infra* Chapter 4. But finding more accurate, distributionally based representations of phrases and entire sentences remains an active area of research. (Mitchell and Lapata, 2010) compared additive models with multiplicative ones for finding similarities among adjective-noun, noun-noun, and verb-object phrases, and found multiplicative models significantly better. Mikolov et al. (2013a) identified words that appear often together and learned vectors for entire phrases. Some researchers have used various

neural networks to combine word vectors into phrase or sentence representations (Socher et al., 2012; Tai et al., 2015). Others have tried to generate specialized word vectors with a focus on compositionality (Kenter et al., 2016). Still others try to generalize techniques from building word embeddings to building embeddings for entire sentences, paragraphs, or documents (Kiros et al., 2015; Le and Mikolov, 2014).

1.1.2 Composing to Sentences

Shallow Semantics Sometimes, representing all of the semantic information in a sentence may be overkill. Shallow semantics refers to efforts to represent just a small portion of semantic information about the sentence. Semantic Role Labeling (SRL) has been a major example of this area. SRL seeks to label constituents within a sentence with their semantic roles—for instance, to identify the *agent* and *patient* of a verb (Gildea, 2002).

If we label "[John $_{\text{AGENT}}$] **ate** [the fish $_{\text{PATIENT}}$]," we see that John is the agent who did the eating. Thus it is apparent that "[The fish $_{\text{AGENT}}$] **ate** [John $_{\text{PATIENT}}$]," means something quite different—the fish did the eating—whereas the syntactically different "[The fish $_{\text{PATIENT}}$] was **eaten** by [John $_{\text{AGENT}}$]," is a paraphrase of the first sentence.

Frame semantics is similar to SRL, but adds a word-sense disabmbiguation component (Das et al., 2014). Frames take into account that different types of verbs will have different types of roles for their arguments. For instance,[4] "eat," "dine," and "devour" all signify the *ingestion* frame, while "hike," "lope," and "mosey" all invoke the *self_motion* frame. *Ingestion* may have an *ingestibles* argument, which signifies the thing being eaten; obviously, *self_motion* would not be expected to have such an argument. Likewise, although *self_motion* may have *path* and *direction* arguments, *ingestion* typically would not; one may "mosey north" but not "eat north." The task of frame-semantic parsing involves identifying predicates that evoke a frame, figuring out which frame the predicate evokes, and

[4]Example frames are from FrameNet, https://framenet.icsi.berkeley.edu/fndrupal/framenet_search

identifying which constituents fill which arguments in the frame. The task relies heavily on hand-generated resources such as FrameNet (Fillmore et al., 2003).

While it is commonly viewed as an information extraction task, relation extraction can also be construed as a form of shallow semantics; indeed, in some ways, it is simply the other side of the frame-semantic parsing coin. Whereas frame semantics focuses on finding a predicate and the arguments that fill its slots, relation extraction involves identifying entities and the relationships between them. Extracted relations may be used to build or expand knowledge bases (Mitchell et al., 2015); other applications include question answering and mining biomedical texts to identify interactions that may help in developing drugs (Bach and Badaskar, 2007).

Representations in the Semantic Parsing Literature Semantic parsing is commonly defined as the task of transforming natural language text into a complete, formal meaning representation (Clarke et al., 2010). Different researchers in this field have used different MRs. Some have focused on general-purpose MRs, such as lambda calculus, Prolog, and AMR, while others aim for more task-specific MRs.

Lambda calculus represents meanings as functions applied to variables and constants (Artzi et al., 2013). For example, the natural language sentence "what states border texas" may be expressed as $\lambda x.state(x) \wedge borders(x, texas)$ (Zettlemoyer and Collins, 2005). Here, x is an example of a variable, while $state()$ is a function that applies to a single entity and $borders()$ is a function describing a relationship between two entities. Breaking the entire expression down, we see that it represents each entity x such that x is a state and x is in a $borders$ relationship with $texas$. Lambda calculus may also use higher-order functions, which are functions that take other functions as their input. It can incorporate quantifiers and represent all of first order logic (Artzi et al., 2013).

Lambda calculus has been a popular MR and the target of many semantic parsing systems (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Artzi and Zettlemoyer,

2013; Reddy et al., 2016), perhaps because of its flexibility and expressiveness. Navigational instructions for a robot, questions about geography, and queries about flight information can all be represented in this language. Yet it is not straightforward to write, making the generation of anything more than toy datasets difficult (Liang et al., 2011; Iyer et al., 2017).

Prolog is a declarative programming language based on first-order logic. Unlike procedural programming languages (such as Python and C++), declarative programming languages describe facts and rules. In Prolog, these facts and rules are represented by relations, and queries may be run over those relations. Prolog is thus not terribly dissimilar from lambda calculus, and it has the added benefit of being a programming language.

Abstract Meaning Representation (AMR) is a recently-developed representation seeking to unify the somewhat fragmented world of semantic annotations (Banarescu et al., 2013). It represents sentences as rooted, directed, graphs, with labels for the edges and leaf nodes. It incorporates PropBank frames, coreference, modality, negation, reification, and other concepts from a variety of views on what is important in semantics. Much of the research on AMR has been about how to parse English sentences to AMR, but recent work also explores its usefulness to such diverse applications as summarization (Liu et al., 2015), headline generation (Takase et al., 2016), biomedical event extraction (Rao et al., 2017), machine comprehension (Sachan and Xing, 2016), and question answering (Mitra and Baral, 2016).

Task-specific MRs are an alternative to general-purpose MRs. Task-specific representations are designed to be executed in order to achieve certain types of goals. For example, CLANG is the coaching language used to advise "players" in the RoboCup soccer simulation (Chen et al., 2003). It is effective at representing instructions and rules relevant to soccer, such as "If our player 4 has the ball, then our player 6 should stay in the left side of our half" (Wong and Mooney, 2006). It has no way of expressing "Who teaches discrete mathematics?" because querying a relational database is irrelevant to the task of coaching

a robot soccer team. Other formal languages represent navigational instructions for robots, such as "Go left to the end of the hall" (Matuszek et al., 2013), but do not include the concept of players or a ball. Similarly, SQL, which is designed to query relational databases, can express the question about discrete mathematics (given a database schema that includes courses and instructors); however, it can express neither where soccer players ought to be based on the state of the game nor which hallway a robot ought to turn down. If-this-then-that (IFTT) recipes (Quirk et al., 2015) are simple conditional programs that connect apps, social media, smart-home devices, and so on using triggers and actions; an example recipe could express the instruction "turn on my lights when I arrive home."

1.1.3 Grounding

While a great deal of effort has gone into accurately representing sentence meanings, it is important to realize a limitation: the forms themselves express relationships among symbols, but they do not indicate what those symbols mean. Language describes things and people and actions and concepts, many of which exist in the real world. Without some grounding, there is no meaningful distinction between *Eats(John, fish)* and *Eats(fish, John)*, or between either of those and *DancesWith(Kevin, wolves)*; each is simply a form of *Predicate(Arg0, Arg1)*. Grounding describes tying MRs to these real things.

Symbols may be grounded in a variety of ways: predicates may be relationships in a knowledge base and arguments the entities in that base (Zelle and Mooney, 1996; Reddy et al., 2014); the symbols might belong to a vocabulary of instructions that cause a robot to execute certain behaviors (Kate et al., 2005); they can be tied to images or video; or they can be based in distributional representations of words (Beltagy et al., 2014). In this subsection, we focus on grounding entities and perceptual grounding.

Resolving Entities and Coreferences An important task in understanding the meaning of a sentence is understanding the entities—people, places, things, ideas, organizations,

and so on—in that sentence. One component of that is coreference resolution; that is, identifying which constituents in one or more sentences refer to the same entity. For example, in the sentence "John ate the fish, and he found it delicious," the entity doing the eating ("John") is the same as the one finding something delicious ("he"), even though the surface forms of the words are different. Coreference may involve a pronoun and its antecedent, as in that example, but it can also involve different noun phrases referring to the same entity, as in, "When Barack and Michelle Obama arrived, the President waved as the First Lady smiled at the crowd." Here, "the President" and "Barack" refer to one entity, while "Michelle Obama" and "the First Lady" refer to another.

Entity resolution is the related task of tying an entity in a sentence to some real-world entity. For instance, in the real world, there is a person named Barack Obama, who is the 44th President of the United States, the husband of Michelle, the father of Sasha and Malia, owner of Bo and Sunny, and so on. We would like to somehow recognize that both "Barack" and "the President" in our example sentence are referring to this real-world person. While on its surface, this might seem like a simple enough task—just match the names in the text to a database of names—it is far from trivial. For one thing, more than one person may have the same name; "John Roberts" may refer to the Chief Justice of the Supreme Court in one context and the actor who voices a character on the animated show *Bob's Burgers* in another. For another, entities may change roles over time: a 2003 article referring to "the President" would likely mean George W. Bush, a 2016 article is probably talking about Barack Obama, and one from 2017 probably means Donald Trump.

An alternative to resolving an entity to a specific real-world entity is to simply identify its type. Many named-entity recognition systems are designed to identify an entity designated by a name as an instance of "Person," "Organization," "Location," and similar categories. The first step of this task may simply be to determine where an entity's name begins and ends. For example, in the sentence, "The story first appeared in the New York Times," a system must determine that "New," "York," and "Times" are all part of the same

organization name, lest it accidentally label only "New York" as a location.

Perceptual Grounding Language may also be grounded in other modalities, such as images, video, audio, or haptics. Human language acquisition remains a much-debated area despite many years of research; however, no one who has ever seen a toddler point at something in the world and announce "doggy" or "big boat" can doubt that grounding in other senses is an important part of it. It should therefore be no surprise that NLP researchers are interested in using other modalities as well.

Feng and Lapata (2010), Bruni et al. (2011), and Roller and Schulte im Walde (2013), for example, learned semantic representations using corpora that included both text and images. Thomason et al. (2017) describe a method for learning verb meanings from videos; for instance, generalizing from videos of overhand and underhand throwing that "throwing" means a hand and an object move together, then the object moves away from the hand. Multimodal grounding uses more than one modality. Thomason et al. (2016) found that a robot was better at playing "I Spy" with a human when it had haptic, auditory, and proprioceptive information in addition to the typically-available visual data.

1.1.4 Natural Language as Its Own MR

If we seek a representation that can completely capture the precise meaning of a piece of natural language text, a reasonable question is, can we simply use natural language itself? Natural language is, after all, the representation humans use to communicate meaning to one another. It is by definition broad enough to represent the meaning of any sentence. Unlike many representations, lay people can understand it. And it may save a great deal of parsing effort (assuming the natural language representation is the same as the original natural language text), as well as the opportunity for errors in the parsing stage to propegate to downstream uses.

The obvious disadvantage is that natural languages are not designed for ease of com-

putation. Unlike MRs such as Prolog or SQL, natural language is not directly executable. Special programs must be written to operate on the NL strings.

Nevertheless, operating on the surface form of natural language strings is a valid option. Natural logic, for example, is "a logic whose vehicle of inference is natural language" (MacCartney and Manning, 2007); that is, it uses rules about the surface forms of language to reason over text without converting it to first order logic or lambda calculus. In addition to the inference and entailment problems for which it was first proposed, natural logic has recently proven useful for question answering (Angeli et al., 2016). Likewise, rule-based systems for text simplification, such as Siddharthan and Mandya (2014), may operate on the surface form of the text.

1.1.5 Applications of Meaning Representations

While natural language understanding is far from a solved problem, work in semantics has already shown promise for text summarization, question answering, reasoning over text, and dialog systems. Moreover, sentence-level semantics representations are important for understanding semantic relatedness and similarity of sentences; this includes entailment, inference, paraphrase, and clustering. And as new problems, such as fake news detection, emerge, meaning representations beyond the text itself may be essential to solving them.

Summarization is the task of conveying the most important information from one or more documents in a shorter form. Summaries are an essential tool for addressing information overload. They may help a human reader decide whether to spend time reading a particular article, book, or report; or they may act as a substitute for reading the entire document. A summary may convey information from a single document or multiple related documents. Most summarization work is extractive—that is, it builds a summary from sentences copied from the original document; only very recently have efforts at abstractive summaries, which involve the generation of new sentences, seen a small amount of success.

Any summarization system that uses the similarity of sentences to identify the important concepts of a document (or group of documents) must rely on some sort of MR. Thus, for example, LexRank (Erkan and Radev, 2004a) and TextRank (Mihalcea and Tarau, 2004), which build weighted graphs of sentences and identify centroids to extract for a summary, must use a meaning representation of those sentences in order to measure their similarity. Traditionally, tf-idf has been used; however, Bawakid and Oussalah (2008) experimented with a similarity metric that used the WordNet ontology.

Recent years have seen a rapid expansion of the use of different MRs for summarization. Khan et al. (2015) incorporated semantic role labeling, and Liu et al. (2015) assembled AMR graphs for individual sentences into a single large summary graph, from which summary text might be generated. MRs based in deep learning have surged in popularity for summarization since Rush et al. (2015)'s introduction of an encoder-decoder model for abstractive summarization. For example, Nema et al. (2017) creates a neural representation of both the text to be summarized and a query to perform query-based summarization, and Ma et al. (2017) incorporates a semantic similarity evaluation component into their encoder-decoder network to compare source texts and generated summaries, avoiding the problem of summaries that are grammatical but lack "semantic relevance" to the text.

Question answering (QA) exists at the intersection of natural language processing and information retrieval. The goal is to find information that answers a user's question. Some systems are designed to retrieve answers from a structured knowledge base; others attempt to extract the information from a natural language corpus, sometimes even from the web. Questions may cover many topics, and they come in many forms, which are often distinguished by the type of answer they require. Factoid questions, such as "When was Barack Obama born?" can be answered in just a few words. Some questions require lists as answers; a correct answer to "What do pigs eat?" should include numerous things. Still other questions, such as "How can I declare a CS major?" seek a procedural answer. The right

MR may be helpful in representing the question, recognizing relevant text that may answer it, and, for non-factoid questions, in generating a readable answer.

Judging by the datasets commonly used for semantic parsing, answering questions seems to be a major application of semantic parsers. The Air Travel Information System (ATIS) (Dahl et al., 1994) consists of spoken questions used to retrieve information about airlines and flights from a relational database. GeoQuery (Zelle and Mooney, 1996) is a set of questions about U.S. geography, paired with Prolog expressions that can be run over the Geobase database of geography facts. The JOBS dataset (Tang and Mooney, 2001) consists of questions related to a database of job opportunities, with first-order logic representations. Some more recent work has sought to address wider domains. Free-917 (Cai and Yates, 2013) consists of 917 questions that can be answered by Freebase and a lambda calculus representation for each question, representing 81 domains and 635 distinct Freebase relations. WebQuestions (Berant et al., 2013) was generated by collecting questions from Google and the correct answers according to Amazon Mechanical Turk workers; only the questions and their correct answers are part of the dataset, and the logical form is considered a latent variable. SpaceBook (Vlachos and Clark, 2014) includes both questions and other dialog acts in discourses between a tourist and a guide.

But the use of MRs for question answering goes far beyond querying knowledge bases. For example, a recent SemEval QA task focuses on community question answering (CQA) (Nakov et al., 2017). On CQA forums, such as Yahoo!Answers and StackOverflow, users post questions and community members post answers. Potential downsides include the time it takes to sort through many answers to find the right one, as well as the risk of wasting community members' time answering the same question more than once. NLP can help solve these problems. If a user is about to ask a question that is quite similar to one that has already been asked, showing the user that question and its answers may obviate a redundant posting; this is a semantic similarity task. A related task is answer-reranking based on similarity between the question and the candidate answers, which may help in

selecting the best answers to present to the user.

Text generation means starting with some information to be conveyed and turning it into a natural language sentence; in some ways, it is like semantic parsing done in reverse. It requires choosing the right words, then modifying and ordering them so that they fit together in grammatical ways. For example, suppose we wish to generate a sentence where the verb is "see," the tense is past, the subject is "John," and the object is "cat"; further suppose we wish to modify the verb with the adverb "briefly" and the object with "grey," "little," and "furry." The appropriate sentence is easy for a human to think of: "John briefly saw a furry little grey cat." Interestingly, if the order of adjectives were changed to "a grey little furry cat," the sentence would sound peculiar to native English speakers. Notice also that the article "a" is appropriate to the noun "cat" because cats are countable; a mass noun, such as "sugar," would not take the same article. Sometimes one sense of a word may be countable and another may not, so the decision to use an article can subtly alter the meaning of the generated sentence: a person who has "skin" is simply a normal human being, but a person who has "a skin" apparently uses animal hides for decorating. Subtleties such as this make transforming a meaning representation into a sentence a non-trivial task. In addition, it may be possible to express the same MR in a number of different sentences; for instance, the same AMR graph may correspond to several syntactically distinct sentences (Flanigan et al., 2016).

Alternatively, the information to be conveyed may be in some other modality. In this case, text generation requires first generating the desired MR from the information source, then transforming that MR into the appropriate surface representation in natural language. Examples include captioning images (Vinyals et al., 2015) and describing videos in text (Venugopalan et al., 2015).

Dialog systems combine several applications of sentence semantics. A dialog system is an agent that interacts with humans through speech. While this can include chat-bots,

a more interesting case is goal-driven dialog agents. In that case, the human wishes to accomplish something by talking with the agent. For instance, a user might ask a smart-phone assistant what restaurants are nearby, then check if one of them is open, and finally ask for directions to it. Dialog systems must have sufficient natural language understanding capability to recognize and answer questions, as well as the ability to generate appropriate natural language responses. The ability to recall conversational state can also be helpful; in our smart-phone example, if the user asks "What other restaurants are nearby?" the assistant needs to remember the previously-suggested restaurant to take the "other" constraint into account.

Project Sapphire is a collaboration between researchers at IBM and University of Michigan to build a task-oriented dialog system that can provide academic advising to undergraduate computer science and engineering students. Such a system will need to understand student statements and questions, ask for clarifications, retrieve relevant information, reason over information from both the conversation and external knowledge bases, and generate appropriate natural language responses, all in real time. Whereas conversational agents today tend to be rule-based, Project Sapphire's planned system is data-driven, requiring collection and annotation of huge amounts of speech and text. This enormous undertaking requires expertise from many specialists, in areas from natural language processing to reinforcement learning to emotion recognition to crowdsourcing. A variety of meaning representations, from SQL queries to Pyke to AMR, might help with portions of this project. Natural language understanding and generation will be essential to the project's success.

Of particular relevance to the present work, the Sapphire Advisor has access to a relational database of courses, instructors, program requirements, and student histories. The ability to parse student utterances into SQL queries would enable the system to answer questions such as, "Who teaches EECS 482 in the fall semester?" Of course, SQL queries return tables, not sentences, so text generation will also be important to turn those tables into dialog.

Moreover, a central dialog manager will need to maintain some representation of the meaning of the conversation so far. Not every student utterance can or should be parsed to SQL. As a simple example, a dialog manager ought to distinguish social utterances such as "How are you?" from substantive questions. A more sophisticated system might be trained to distinguish a question requiring a database search from one requiring reasoning, such as "What courses do I need to take before EECS 482?"

1.1.6 Semantic Relationships between Sentences

Several interesting tasks involve understanding the relationship between the meaning of one sentence and the meaning of another.

Semantic relatedness is the task of predicting how similar humans think two sentences are (Marelli et al., 2014a).[5] Two sentences that have essentially nothing to do with each other should receive a score of 1; two sentences that mean exactly the same thing should receive a 5.

Paraphrase detection is a similar, but binary, problem: do these two sentences mean the same thing? Strictly, two sentences A and B are paraphrases if knowing A to be true means B must be true, and knowing B to be true means A must be true. The task can be more difficult than it seems. As we have seen repeatedly from the "John ate the fish" and "The fish ate John" example, simply having the same words does not mean two sentences mean the same thing. Issues such as negation scope may arise, as in "The boy who was not wearing dark glasses could see the man" and "The boy who was wearing dark glasses could not see the man." Different words might express the same thing, as in "Jane spoke with her uncle and his son" and "Jane talked to her cousin and his father."

[5] We adopt the term "semantic relatedness" for this task because that is commonly used in the literature. In reality, that phrase is perhaps overly broad for what is really a measure of semantic similarity only. Semantic relatedness might reasonably be taken to encompass all of the relationships between sentences described in this subsection; however, here we use it strictly to refer to this one task.

Or one sentence may contain additional information, as in "Athena was the goddess of wisdom, who sprang forth fully formed from her father's head," and "Athena was the goddess of wisdom." In this case, if sentence A is true, sentence B must be true, while knowing sentence B to be true does not mean sentence A must be true. This example is not, then, a paraphrase. It is, however, an entailment.

Recognizing textual entailment (RTE) is a task that takes as input a text, T, and a hypothesis, H. If knowing T to be true means H must be true, then T is said to entail H. Sometimes RTE tasks also ask about other relationships between T and H; for example, if T's truth means H must be false, then the pair is a contradiction.

Presuppositions are facts that a sentence *assumes* rather than *asserts*. For instance, "Sasha's brother enjoys mangoes," presupposes that Sasha has a brother and asserts that person's enjoyment of mangoes. If a presupposition of a sentence is false, the sentence is not true—philosophers debate whether it is false or demonstrates some third truth state. Like entailment, recognizing presuppositions can give us more information about the world than what the sentence directly asserts, making it potentially useful for tasks such as information extraction. Unlike entailments, however, presuppositions survive many modifications to the original sentence, including negation ("Sasha's brother does not enjoy mangoes."), changes to modality ("Sasha's brother might enjoy mangoes if he would only try them."), and making the sentence a question ("Does Sasha's brother enjoy mangoes?").

1.2 Transforming Text to MRs

Once an appropriate MR has been selected, how can we transform natural language text into that MR? It depends upon the MR. If we choose a simple representation such as word-count vectors, the algorithm is straightforward. However, for more sophisticated representations,

such as lambda calculus and SQL, the problem is far from solved. Related work has made inroads, as described in this section, but significant room for improvement remains.

1.2.1 Semantic Parsing

Semantic parsing means taking as input a natural language sentence and generating as output a meaning representation of that sentence. As we saw in Section 1.1.1, "meaning representation" (MR) is a broad term, encompassing many possible embodiments of the information in the sentence that enable a machine to operate on the information. Existing work covers a wide variety of MRs, including lambda calculus (Zettlemoyer and Collins, 2005), dependency-based compositional semantics (Liang et al., 2011), Lambda Dependency-Based Compositional Semantics (Liang, 2013), probabilistic logic (Beltagy et al., 2014), Abstract Meaning Representation (AMR) (Banarescu et al., 2013), semantic graphs (Reddy et al., 2014; Yih et al., 2015), frame semantic representations (Das et al., 2010; Johannsen et al., 2015), a language incorporating both predicates and dialog acts (Vlachos and Clark, 2014), SQL (Iyer et al., 2017), and logical forms to be executed on a knowledge base (KB) (Berant and Liang, 2014).

Producing a high-quality meaning representation should facilitate applications that benefit from some degree of natural language understanding. Berant and Liang (2014) demonstrated improvements in question answering and information extraction using their semantic parser; Beltagy et al. (2014)'s semantic parser improved recognition of textual similarity and textual entailment. Vlachos and Clark (2014) developed a meaning representation for a dialog system.

Many different techniques have been explored for semantic parsing. An exhaustive review of all of them is beyond the scope of this introduction; rather, we identify a few of the large trends.

One very popular and successful approach has featured combinatory categorial grammars (CCGs) (Bos et al., 2004; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins,

2007; Kwiatkowski et al., 2010; Artzi and Zettlemoyer, 2011; Kwiatkowski et al., 2011; Artzi and Zettlemoyer, 2013; Kwiatkowski et al., 2013; Reddy et al., 2014; Krishnamurthy and Mitchell, 2014; Wang et al., 2014a; Artzi et al., 2015; Misra and Artzi, 2016). A CCG has a lexicon of words and their syntactic types, and CCGs used for semantic parsing also have a semantic component. For example, a simple grammar might include the lexicon in Table 1.1.

dogs	NP	*dogs*
cats	NP	*cats*
like	(S\NP)/NP	$\lambda x.\lambda y.like(x, y)$
play	S\NP	$\lambda x.play(x)$

Table 1.1: A simple CCG lexicon.

Syntactic types encode information about how different words combine. The syntactic type *S\NP* for the word "play" means that when a word with the syntactic type NP is to the left of "play," they can combine to form a phrase with the syntactic type S. So "dogs play" can be parsed to a sentence, S. The syntactic type *(S\NP)/NP* for "like" indicates that it can be combined with an NP to its right to form a phrase with syntactic type S\NP, which can in turn be combined with an NP to its left to form a sentence. So we can combine "like" with "cats" to give us a phrase "like cats" with a type of S\NP, and we can then combine "dogs" with that phrase to give us "dogs like cats" with a type of S.

CCG-based semantic parsers include a logical form component in each word's lexicon entry. Thus, when we combine "dogs" and "play," we also combine their semantic components, *dogs* and $\lambda x.play(x)$ to get *play(dogs)*. When we combine "like" with "cats" we get $\lambda x.like(x, cats)$, and when we combine that with dogs, we get *like(dogs, cats)*.

One challenge of CCG-based semantic parsing is creation of resources. Lexicons are frequently domain-specific; for example, the one for the GeoQuery domain will not contain terms like "airport" and "fly" that are essential to the ATIS air travel domain. If a parser requires a CCG lexicon, porting it to new domains may require a great deal of human effort to build a lexicon for the new domain.

Classical NLP methods based in parsing and grammars have had and continue to have a large role in semantic parsing. Shift-reduce parsing was an early technique (Zelle and Mooney, 1996) but remains relevant (Zhao and Huang, 2015; Misra and Artzi, 2016). Damonte et al. (2017) describe a transition-based semantic parser. Several researchers have used synchronous context-free grammars (Wong and Mooney, 2007; Arthur et al., 2015; Li et al., 2015), while others have used methods focused on the syntactic structure of the sentence (Ge and Mooney, 2009) or dependency trees (Reddy et al., 2016).

Labeling training data is challenging for many NLP tasks, but this is particularly true for labeling sentences with their meaning representations, as meaning representations are rarely intuitive to human annotators. A number of approaches attempt to reduce reliance on annotated examples, by treating the MR as a latent variable (Berant et al., 2013), or by using distant supervision (Parikh et al., 2015) or unsupervised approaches (Goldwasser and Reichart, 2011).

Some techniques make use of the fact that a single meaning representation might serve for a number of sentences that express the same idea. If it can be determined that a new sentence is sufficiently similar to a canonical sentence, the canonical sentence's MR may be applied to the new sentence. Paraphrase and sentence rewriting are thus being explored. (Berant and Liang, 2014; Chen et al., 2016)

Andreas et al. (2013) described semantic parsing as a translation problem and suggested applying machine translation techniques to solve it. Dong and Lapata (2016)'s neural network approach also grows from machine translation techniques.

1.2.2 Deep Learning Approaches

Neural networks have lately become a very popular tool for NLP, and they have recently been applied to semantic parsing (Grefenstette et al., 2014; Dong and Lapata, 2016; Jia and Liang, 2016).

Most use of neural networks for semantic parsing have focused on sequence-to-sequence

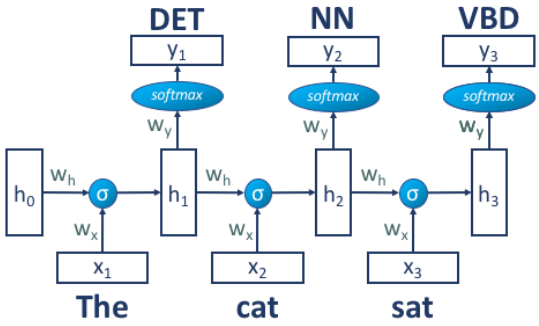


Figure 1.1: An example recurrent neural network for part-of-speech tagging.

(seq2seq) models (Sutskever et al., 2014). Developed for machine translation, seq2seq models are typically made of two recurrent neural networks: an encoder and a decoder.

Recurrent neural networks (RNNs) are a deep learning architecture made of a sequence of cells to reflect a sequence of inputs, as shown in Figure 1.1. In NLP, these inputs are usually words. At each timestep in the sequence, the input to the cell comes from two sources: the word at that timestep and the hidden state of the cell at the previous timestep. The model applies one set of learned weights (w_x in Fig. 1.1) to the input word vector and another set (w_h in Fig. 1.1) to the hidden state that is being passed forward, then combines them and applies a nonlinear function, such as sigmoid or tanh. The hidden state of each cell may be used to emit output at each timestep; in this example, we show still another set of learned weights (w_y) applied to the output at each timestep, and the result being passed through a softmax classifier to predict the part of speech for each word in the sentence. Alternatively or additionally, the hidden state of the final cell in the network may be passed to a classifier.[6] The final hidden state encodes information about the entire sentence that has been passed forward through the sequence of cells. Long Short Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are a form of RNN with cells that use memory gates to avoid some difficulties that can arise in training RNNs on long sequences.

In a seq2seq model (Sutskever et al., 2014), like the one shown in Figure 1.2, the first

[6]The baseline system in Chapter 5 illustrates an example of a model that emits output at each timestep and at the end of the sequence.

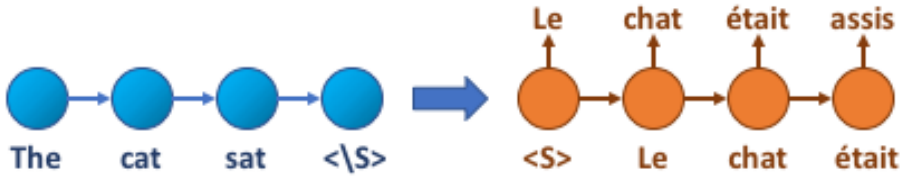


Figure 1.2: An example seq2seq model for machine translation. Blue circles are cells of the encoder and orange are cells of the decoder.

sequence is encoded by one LSTM. The hidden state from that encoder is then passed forward into the second LSTM, the decoder. The decoder emits a prediction at each timestep for the most likely token in a sequence. For example, a seq2seq translation model might read English text into the encoder and attempt to emit French from the decoder, as in Figure 1.2. Each cell of the decoder receives as input the hidden state of the previous cell and the token that the previous cell predicted. The basic idea for use of seq2seq models for semantic parsing is that, rather than learning to emit a French representation of an English sentence, the model should be trained to emit an MR, such as Prolog, lambda calculus, or SQL.

Such models require a great deal of training data. For the translation example, this means aligned parallel text in English and French. For semantic parsing, it means pairs of natural language sentences and their MRs.

Dong and Lapata (2016) used a seq2seq model with attention and a related sequence-to-tree model for semantic parsing to logical forms. Kočiský et al. (2016) presented a similar but semi-supervised model. Jia and Liang (2016) described a seq2seq attention-based copying, which could copy words from the input English sentence to the output logical form; they also used data recombination to overcome the limited supply of labeled English/logical form pairs. Iyer et al. (2017) applied seq2seq models with some automatic dataset augmentation to semantic parsing to SQL. Due to the larger number of people who know SQL than logical forms, they were able to obtain user feedback, permitting them to train a model in a new domain. Zhong et al. (2017) avoided the dataset bottleneck by generating a new dataset of hundreds of thousands of questions and corresponding SQL

queries; unfortunately, they are all very simple queries with no joins between tables or nested subqueries.[7] They also incorporate a representation of the database schema into their network. Noord and Bos (2017) describes a method of neural semantic parsing to AMR that helps deal with coreference.

Doubtless this area will grow rapidly, at least in the near term. A search of arxiv[8] reveals nearly four hundred preprints mentioning "neural semantic parsing" as of this writing. We can hope to see changes in network architecture to better reflect the types of MR being output, as well as more varied large datasets or improved ways of coping with limited amounts of data.

1.3 The Present Work

The remainder of this thesis is divided into two parts. In the first, we consider questions of how we should represent meaning. In the second, we choose a fixed meaning representation—here, we use SQL—and examine methods of parsing short texts from English into that representation.

1.3.1 What Meaning Representations Should We Use?

The first question to ask when choosing a meaning representation is, why bother changing the text at all? Humans use natural language to represent meaning. Why not simply program computers to work with the original, natural language text? Surely that is easier than transforming the text into some other representation, performing the desired operations on that representation, and then transforming the MR back into text.

Chapter 2 demonstrates this approach on an automatic summarization problem. As noted above, most successful automatic summarization systems are extractive—they select

[7] See Chapter 5 for discussion of why these are important.
[8] https://arxiv.org/

sentences from the original text and piece them together to create a summary. Yet the sentences in a document are not written with an eye to being part of a summary. A single sentence may contain both information that is important enough to belong in a summary and extraneous facts that do little but take up valuable space. This is particularly true in contexts where sentences tend to be long, containing many clauses, as is the case in sophisticated documents, such as legal cases. It would be nice to rewrite each long, complex sentence as a few shorter, simpler sentences that retain the meaning of the original; then, extractive summarization techniques can select only those of the shorter sentences that are actually important. We compare three techniques that do this: sentence compression, sentence simplification, and sentence disaggregation. We discover, though, that manipulating the text itself according to a set of rules is problematic. This suggests the necessity of representing sentence meaning in other ways.

Next, in Chapter 3, we explore neural MRs to help detect paraphrases. We compare the performance of traditional, sequential LSTMs with two alternative architectures: siamese LSTMs and siamese dependency tree LSTMs. A traditional LSTM takes a single sequence as input; for paraphrase detection, this means inputting the first sentence, followed by a delimiter token, followed by the second sentence; the final hidden state of the LSTM is then fed into a softmax classifier to classify the pair of sentences as paraphrase or not. In a siamese LSTM, two sequential LSTMs run, one on each sentence, using the same parameters. These generate a hidden representation of each sentence, which can then be compared. A dependency tree LSTM structures the LSTM cells according to the dependency parse of a sentence, rather than sequentially. Surprisingly, we found no significant difference in performance among these three approaches.

Then, in Chapter 4, we focus on choosing an MR for short text clustering. We show that, in fact, there is not a single "right" MR for clustering, even though it sounds like a single task. Interactions between characteristics of the corpus we wish to cluster and the clustering algorithm we choose to apply make selection of an MR a complex issue. We propose

a measure for the creativity of a clustering corpus based on vocabulary width. Logically, it seems that if authors try to express themselves in a variety of ways, word-count and tf-idf representations will not be able to capture the similarity in meaning between different texts, because they have no representation of the similarity between words. In either representation, "dog" and "canine" are as completely different as "dog" and "airplane." Neural and distributional MRs, in contrast, may incorporate information about word similarity. And, indeed, when we use the common k-means clustering algorithm, we find a benefit to neural and distributional MRs on the most creative dataset. However, when we vary the clustering algorithm, the tf-idf and word-count MRs had a decided advantage: they produced tightly clustered data, compared to the more loosely clustered neural and distributional options. While cluster tightness had little effect on k-means, it had the ability to make or break many powerful, graph-based clustering algorithms.

Thus, we show that choosing an MR is a complex problem. Answers that seem obvious—using text as its own MR, or modifying neural architectures in ways that should lead to better MRs—do not always work as expected. And the best MR for even a single task varies depending on circumstances.

1.3.2 How Can We Transform Text to a Meaning Representation?

In Part II, we delve into how to generate a selected MR from text. For this section, we use SQL as the MR. SQL (short for Structured Query Language) is a programming language used to query relational databases. Relational databases are a popular method of storing information, making SQL a very useful MR for certain types of information retrieval tasks. We explore the task of transforming English text to SQL queries from three angles: methodological, system-building, and application.

We begin with methodology in Chapter 5 by taking a hard look at how the text-to-SQL task has been evaluated in the past. There are two important decisions in how to evaluate a text-to-SQL system: which datasets to use and how to divide them into train, development

(dev), and test sets. We present a number of complexity measures for text-to-SQL datasets. Using these measures, we compare several text-to-SQL datasets. We show how human-written datasets reflect a type of complexity that automatically-generated datasets lack. This suggests that either new methods for automatically generating datasets are needed, or systems ought to be evaluated on a variety of datasets to ensure that they are capable of handling complex queries. We introduce a new text-to-SQL dataset that incorporates multiply nested queries and joins.

We then show how performance on traditional dataset splits fails to tell us about a system's generalizability to new meanings. Traditional splits treat each English question and corresponding SQL query as an example. No example may appear in both the training data and the test data, of course. However, many questions within datasets are paraphrases of one another; for instance, the traditional split of the GeoQuery dataset includes "how big is Texas" in the training data and "how large is Texas" in the test data. The SQL queries for these two questions are identical. Other questions, such as "how large is Alaska," are identical but for the named entities in them. We refer to this traditional split as the *question-based split*, since no English question may belong to both train and test sets. We introduce a new *query-based split*, where no SQL query may belong to both train and test.

When the same query appears in training data and test data, we cannot say that a system's success in generating the correct output query indicates any ability to generate new SQL. The model may instead be learning to classify new input questions according to which previously-seen query they correspond to. If two queries that are the same but for their named entities appear in the train and test respectively, the system's success could indicate that it has learned to classify query templates and fill the slots in those templates. We show that a baseline model that does precisely this achieves near-state-of-the-art performance on question-based splits across four datasets. On the query-based split, however, such a baseline cannot get a single query correct. State-of-the-art models, too, struggle with the query-based split; however, the instances where they do succeed help us to understand

what might help future models generalize to new queries.

In Chapter 6, we propose a system for generating a SQL representation of an English question. Building on previous work that has shown the effectiveness of seq2seq models using attention-based copying from the input sentence, we propose a method that also pays attention to the database schema. SQL queries are specific to a particular database, with particular tables, fields, and relationships among them. The same question can be represented in different ways for databases with different schemas. Thus, any successful text-to-SQL system must incorporate knowledge of the database schema. Typically, a seq2seq model acquires this knowledge implicitly by learning from hundreds or preferably thousands of training examples that share a schema. If the schema changes, though, such a model will not work. Moreover, it means that a model for a new domain needs enormous numbers of training examples within the new domain; it cannot learn about the SQL language in the advising domain and generalize to the geography domain. Our proposed system addresses this problem by providing the model with an explicit representation of the database schema. Such a representation can be easily generated from the database itself. We modify attention models previously used for attention to the input sentence so that they enable attention to the schema representation as well. We report the results of experiments with two such representations, and we propose alterations to both the representation and the network architecture that hold promise for future work.

Finally, in Chapter 7, we investigate how a text-to-SQL model developed with single-sentence English questions can be modified for incorporation into a dialog system. Previous work on text-to-SQL has focused on transforming a single utterance—like *"Who teaches EECS 281?"*—into a SQL query. In a dialog, the information required to generate a complete SQL query may appear across several utterances. As a simple example, consider a two-line dialog:

ADVISOR: You could take EECS 281.

STUDENT: Who teaches that?

A text-to-SQL system could not be expected to generate correct SQL given only the student's utterance, since "that" refers to "EECS 281," an entity that was only mentioned in the advisor's utterance. At the same time, merely feeding the entire conversation to the encoder of a seq2seq model may be problematic. The LSTM-based encoders typically used in such models are not necessarily optimal for very long input sequences.

Even though research into text-to-SQL models has picked up rapidly in the past two years, and dialog systems are a rapidly-expanding application for NLP, to our knowledge no one has begun to explore how the two can fit together. We therefore present preliminary work in this area. We describe the development of a new dataset for the task. We report on a series of experiments aimed to answer the following questions:

- How does a model trained on single-sentence examples from the same domain perform on examples from dialogs?
- Does a dialog-to-SQL model benefit from including single-sentence examples in its training set?

We present our findings as a baseline for this new task, and we propose directions for future work.

By using SQL as an example meaning representation, this thesis is able to explore three different views of the generation of MRs from text. The questions we address here, though, are larger than text-to-SQL. Regardless of the MR selected—and as we saw in Part I, there are many options—research needs to consider methodological questions: Are the evaluations we use telling us what we need to know about the performance of systems on this task? Or do evaluation techniques adapted from other representations overlook factors unique to this task? Likewise, when we build systems, we need to consider whether the representation should inform our system architecture. And we must remember that, ultimately, these systems are meant to be used; we must not lose sight of how they will need to be adapted for applications.

Part I

Selecting a Meaning Representation

CHAPTER 2

Sentence Compression, Simplification, and Disaggregation for Sophisticated Texts

Most automatic summarizers are extractive; they select complete sentences from the original document(s). (Mani and Maybury, 1999; Radev et al., 2002) However, sentences in an original document are often not ideally suited to a summary. One problem is that a goal of summaries is to convey information concisely, but sentences from the original document were not written with this limitation in mind. Thus, an extracted sentence might contain both essential and extraneous information. Several researchers have sought to address this problem through sentence simplification and sentence compression. (Knight and Marcu, 2002; Siddharthan et al., 2004; Zajic et al., 2007)

These efforts have largely focused on deleting portions of sentences deemed unimportant. Even systems that implement forms of simplification and compression other than deletion also use deletion as one of their operations. Although these systems have as their goal the reduction of sentence length with minimal loss of meaning, deletion necessarily entails some loss of meaning.

Naturally, a summary cannot include all of the meaning of the original document. Summarization algorithms use information from the entire document to select important portions of the meaning for inclusion in the summary. It makes little sense, then, to make deletion a separate step from sentence selection, as this takes the decision about what meaning is important away from the summarization algorithm. We might thus expect improved

summarization if we divide the sentences into smaller units of meaning that the selection algorithm can act upon.

Our approach to such division is to disaggregate sentences—that is, split one long sentence into two or more shorter sentences—before running the sentence selection algorithm. As an example, the sentence

> *The district court's decision cannot be affirmed on the ground that the petition was untimely, and we must take up the merits.* (1)

might be disaggregated to

> *The district court's decision cannot be affirmed on the ground that the petition was untimely.*
> *We must take up the merits.* (2)

Disaggregating sentences thus allows extractive summarizers to act on shorter units. Rather than deciding whether all 23 words of Example (1) should be included in the summary, the system can make separate decisions about the 15-word first sentence and the six-word second sentence of Example (2), perhaps allowing it to include important information in the summary while leaving out less important information. Unlike sentence compression, however, disaggregation only *reorganizes* meaning; it does not *remove* content.

Another problem with prior work on sentence compression for summarization is its focus on newswire articles. While research has begun on summarizing documents such as scientific journal articles, legal cases, and dissertations (Jha et al., 2015; Hachey and Grover, 2005; Ou et al., 2007), research on sentence compression for summarization has largely ignored such documents. These sophisticated documents—written for audiences with years of specialized education, like lawyers, scientists, and doctors—are particularly likely to include long, complex sentences, and thus particularly likely to benefit from a procedure that allows the summarizer to choose from shorter units of meaning.

In addition, preserving as much meaning as possible is especially important in these sophisticated domains. For example, while it might be safe to compress

Sources say that the House is likely to vote on the issue tomorrow. (3)

in a newswire article into

The House is likely to vote on the issue tomorrow. (4)

we cannot compress

Plaintiffs say that we must find Defendant in contempt. (5)

in a legal case into

We must find Defendant in contempt. (6)

without significantly altering the meaning. Disaggregation thus seems likely to be particularly useful in sophisticated domains.

We evaluate several alternative methods of summarizing two types of sophisticated documents: legal cases and biomedical research articles. We compare the performance of an extractive summarizer when it selects from unaltered sentences with its performance on simplified, compressed, or disaggregated sentences.

The rest of this chapter is organized as follows. We first describe related work in summarization and simplifying and compressing sentences. Second, we describe the task and the documents we seek to summarize. Third, we describe the existing sentence compression and sentence simplification systems we test with our summarizer, as well as a sentence disaggregation technique that uses 112 manually written rules to split sentences. Fourth, we describe the extractive summarization algorithm used in all conditions of our experiment. Next, we describe our experiment comparing the four systems. Finally, we discuss possible reasons for the observed results, address future avenues of research, and present our conclusions.

2.1 Related Work

2.1.1 Summarization

From the earliest days of automatic text summarization through today, extraction of sentences from the original document has been the preferred approach; see, *e.g.*, (Luhn, 1958; Barzilay and Elhadad, 1997; Hovy and Lin, 1998; Erkan and Radev, 2004b; Mihalcea and Tarau, 2004; Yang and Wang, 2008; Haghighi and Vanderwende, 2009; Piwowarski et al., 2012).

Evaluation of summaries may be intrinsic or extrinsic. Rath et al. (1961) began the popular approach of intrinsic evaluation—that is, of evaluating summaries by determining how similar they were to human-written summaries. Many modern automatic summary evaluations use the ROUGE system (Lin, 2004), which measures the similarity of the automatic summary to several human-written summaries. Similarly, for the pyramid method of evaluation, human judges compare the target summary to several human-generated models. (Nenkova et al., 2007)

An alternative approach is extrinsic (task-based) evaluation. Rather than asking, "How similar is this summary to a human-written one?" we ask, "How well does this summary enable its readers to complete a task?" Morris et al. (1992) evaluated summaries by having subjects who read a condensed form complete a reading comprehension test to assess how well they understood the content of the original document. Hand (1997) described how well humans who read the summaries could categorize the document and decide whether it was relevant to a query. McKeown et al. (2005) tested the ability of human readers to complete a time-limited fact-gathering task using summaries. Otterbacher et al. (2008) used comprehension questions to evaluate the usefulness of hierarchical summaries to readers using mobile devices. Murray et al. (2009) evaluated summaries of meetings by asking human readers about how and why a decision was made at the meetings. And Rastkar et al. (2014) used task-based evaluation on summaries of software bug reports.

2.1.2 Simplification, Compression, and Disaggregation

2.1.2.1 Background and Early Work

Work in shortening sentences falls into two main categories: sentence simplification and sentence compression. Early simplification research was intended to benefit readability, parsing, and summarization and to improve accessibility for people with disabilities. It generally involved simplifying both structure and word choice. Carroll et al. (1998) described a pipeline incorporating analysis, lexical simplification, and syntactic simplification. Their syntactic simplifier was based on handwritten rules, such as replacing passive constructions with active constructions. Chandrasekar and Srinivas (1997) learned simplification rules automatically. In the earliest work on pure sentence compression that we are aware of, Grefenstette (1998) described "telegraphic text reduction" to allow blind people using a reading machine to skim text; a user could choose, for example, to seee only proper nouns; or only subjects, head verbs, and objects of the main clause; or only subjects and object nouns including subclauses.

In the early 2000s, simplification work split in two directions. While some researchers focused on simplification for readability, others began to work on sentence compression for summarization. Confusingly, some work on sentence compression was still called simplification. For consistency, we refer to efforts to shorten sentences by discarding some content as compression, efforts to make text easier to read as simplification, and efforts to split sentences as disaggregation.

2.1.2.2 Sentence Compression for Summarization

Knight and Marcu (2002) presented two approaches to sentence compression for summarization: a noisy-channel model and a decision-tree approach. In both cases, their goal was to generate a grammatically correct compression that included the most important pieces of information from the original sentence but deleted some subset of the words. Turner and

Charniak (2005) created unsupervised and semi-supervised models to complement Knight and Marcu's supervised learning approach. Examining Knight and Marcu's hypothesis that sentence compression could improve summarization, Lin (2003) found that an oracle method of compression—reranking candidate compressions using the manual summaries of the same documents—did improve performance on a forerunner of the ROUGE evaluation (Lin, 2004); however, Knight and Marcu's noisy-channel compression model actually worsened performance. Lin noted that even the oracle condition did not improve summaries as much as expected and suggested that sentence compression "might drop some important content."

Siddharthan et al. (2004) compressed sentences for summaries by removing parentheticals. They found that this improved clustering, in that it got rid of background information. They extracted the desired sentences for the summary, then added the parenthetical information back in where it was needed—the first time the entity it described was mentioned in the summary.

Some groups permitted the sentence extraction module to choose from more than one possible variation on the same sentence. Zajic et al. (2007) altered a parse-and-trim compression approach so that it produced multiple compressions of a sentence, then used a sentence selector to choose from the pool of candidate sentences based on a linear combination of features. Similarly, Vanderwende et al. (2007) wrote manual rules to remove certain syntactic units such as appositives, then provided their summarizer with both the compressed sentence and the original sentence. They relied upon the sentence selection algorithm's ability to deal with redundancy to ensure that it did not select both versions of a single sentence. On DUC 2006 data, over 40% of the sentences the selection component chose to include were the simplified sentences, which resulted in the ability to add one extra sentence to each summary on average.

Using Integer Linear Programming (ILP) for sentence compression (Clarke and Lapata, 2007; Clarke and Lapata, 2008) led to improved results and a burst of related research.

Clarke and Lapata's model maximized a scoring function while adhering to sentence-level constraints to ensure grammatical output. They later added the further step of considering the context surrounding the sentence to be compressed when choosing a compression. (Clarke and Lapata, 2010). Martins and Smith (2009) improved summarization performance by optimizing an objective function with a single set of constraints that incorporated both compression and extraction. Similarly, Berg-Kirkpatrick et al. (2011) found that jointly learning a model for extraction and compression outperforms the model that only learns extraction for multi-document summaries. One problem with their model, was that solving the ILP for joint extraction and compression was an order of magnitude slower than solving the ILP for extraction only; sometimes the process was prohibitively slow. Chali and Hasan (2012) compared three methods of query-focused multi-document summarization using ILP-based compression and extraction: choosing sentence compressions and then doing sentence extraction; doing extraction and then compressing the sentences; and combining compression and extraction. Combined extraction and compression performed the best, followed by first extracting and then compressing sentences. Li et al. (2013), noting that joint compression and selection by ILP can be prohibitively expensive, proposed an alternative: use summary-guided compression to get a set of good possible compressions, apply a pre-selection step, then use the ILP selection framework to select the compressed sentences.

The most recent work with ILP for sentence compression has been in Joint Structural Inference (Thadani and McKeown, 2013). Whereas earlier ILP compression systems used either a language model or edges in a dependency graph to represent text, this approach uses both, providing performance gains without requiring hand-picked constraints. A follow-up to that paper (Thadani, 2014) addressed the slowness of joint inference by solving the two sub-problems separately and generating approximate solutions to the graph-based sub-problem.

2.1.2.3 Simplification for Readability

Work on simplification for readability has continued in parallel. Siddharthan (2002; Siddharthan (2006) proposed a manual-rule-based model with analysis, transformation, and regeneration stages. He described handwritten rules to recognize and simplify adjectival or relative clauses, adverbial clauses, coordinate clauses, subordinate clauses, correlated clauses, participial phrases, appositive phrases, and passive voice based on chunking, POS identification, and constituency parsing. Siddharthan (2010) incorporated handwritten rules that matched patterns in a typed dependency parse and that could make certain lexico-syntactic substitutions so that the resulting sentences included the correct parts POS tags.

Wubben et al. (2012) moved away from the handwritten rules model of simplification and instead used a monolingual Phrase-Based Machine Translation (PBMT) model trained using Wikipedia and Simple Wikipedia as parallel corpora. Noting that simplification work was becoming divided between manual rules and statistical methods, Siddharthan and Mandya (2014) introduced a method that used both manual rules and rules learned from parallel English Wikipedia and Simple English Wikipedia texts.

2.1.2.4 Disaggregation

The only research that we are aware of that disaggregates sentences is Siddharthan (2002; Siddharthan (2006) and Klebanov et al. (2004).[1] While Siddharthan (2002) described a wide range of syntactic constructs that could be simplified according to rules, the rules actually implemented only split certain constructs into multiple sentences. Jonnalagadda and Gonzalez (2010) successfully applied Siddharthan's sentence splitting system to improve recall without harming precision in information extraction from biomedical papers. Klebanov et al. (2004) introduced the concept of an "Easy Access Sentence" (EAS) for

[1] Although termed "simplification" by their authors, these works actually broke long sentences into pieces while preserving the meaning and thus are disaggregation as defined here.

information extraction. An EAS should have one finite verb, make only claims that were present in the original sentence, and include as many entity names as possible. To generate EASes, they used a named entity recognizer and a parser, then constructed a sentence for every verb in the original sentence.

2.2 Problem Definition and Algorithms

2.2.1 Task Definition

Our goal is to generate improved summaries of sophisticated documents by selecting from shorter sentences than the original document contained. By sophisticated documents, we mean documents written for a highly educated audience, which are likely to include long or complicated sentences. For this study, we are considering legal cases and biomedical articles.

In the United States legal system, when a court decides an issue of law, a judge writes a document called an "opinion," "decision," or "case," explaining the circumstances of the case, the issues before the court, and what the court decided and why. Such legal opinions are precedent that guide—and in some situations bind—courts that consider similar issues in the future. Thus, legal professionals must know about opinions. Yet these professionals face information overload, as tens of thousands of opinions are issued in the U.S. federal courts alone each year. Summarization of cases is therefore essential. Unfortunately, legal cases are quite challenging to summarize, in part because of the writing style they often use, where long, complicated sentences are the norm. (Tiersma, 1999)

We assembled a corpus of 30 random legal cases from all thirteen U.S. Federal Circuit Courts of Appeal (2009–2013). The cases include criminal, civil, and bankruptcy matters; administrative agency appeals; and proceedings originating in the appellate courts. No unpublished cases were included. We also collected manually written summaries of each

of these cases from LexisNexis.[2]

Like legal cases, biomedical articles are a domain where there is "a pressing need for distillation and management of complex information presented in vast amounts of text," according to the TAC 2014 Biomedical Summarization Track.[3] We assembled 35 randomly chosen articles from PubMed Central (PMC) (2009–2013).[4]. Topics range widely; our corpus includes behavioral genetics, a thyroid cancer case study, evaluation of the safety of a treatment for stroke, and mapping of a gene.

At first glance, sentence lengths in both the biomedical and the legal corpus appear modest. The development (dev) set for legal cases had a mean sentence length of 24.01 words, while the biomedical articles dev set had a mean of 23.53. For comparison, the corpus of newswire articles for Task 2 of DUC 2004 had a mean sentence length of 25.11. However, the legal and biomedical corpora had much higher variance than the DUC example. Table 2.1 shows the percentage of sentences from each corpus with more than 50 words or more than 75 words. Such long sentences are rare in newswire articles, but common in the sophisticated documents. illustrate, the legal cases and biomedical articles include an unusually high number of extremely short sentences[5] and a long tail of sentences over 100 words long, with maxima over 250 words. Clearly, including a single 250-word sentence in a summary leaves little room for other sentences.

	Sentences Over Length	
Document Type	**50 Words**	**75 Words**
Legal	9.4%	2.0%
Biomedical	8.2%	1.4%
DUC	1.8%	0.1%

Table 2.1: Percentage of sentences from the legal and biomedical dev sets and the DUC comparison corpus containing more than 50 or more than 75 words

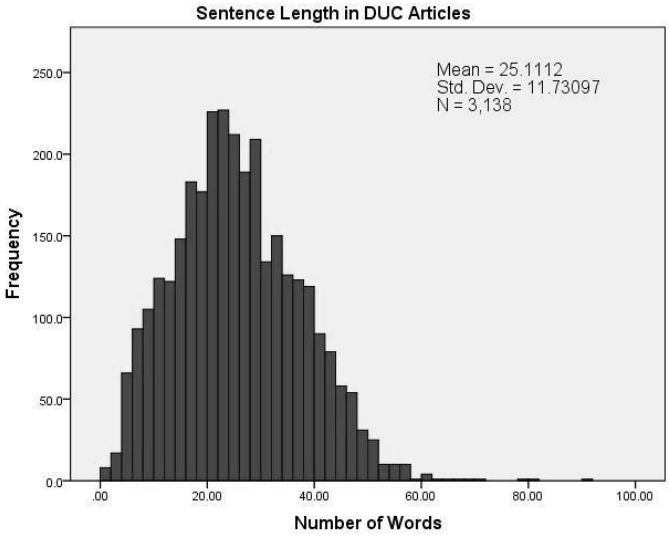
The desired output for this task is a single-document summary of limited length for

[2]www.lexisnexis.com

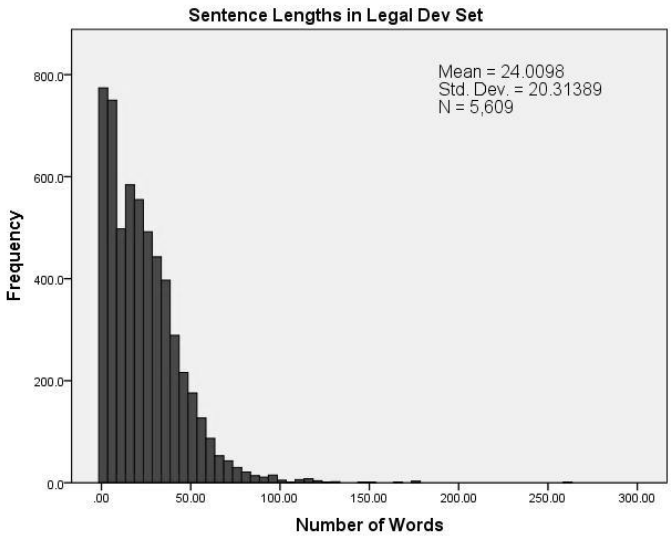
[3]http://www.nist.gov/tac/2014/BiomedSumm/

[4]http://www.ncbi.nlm.nih.gov/pmc/

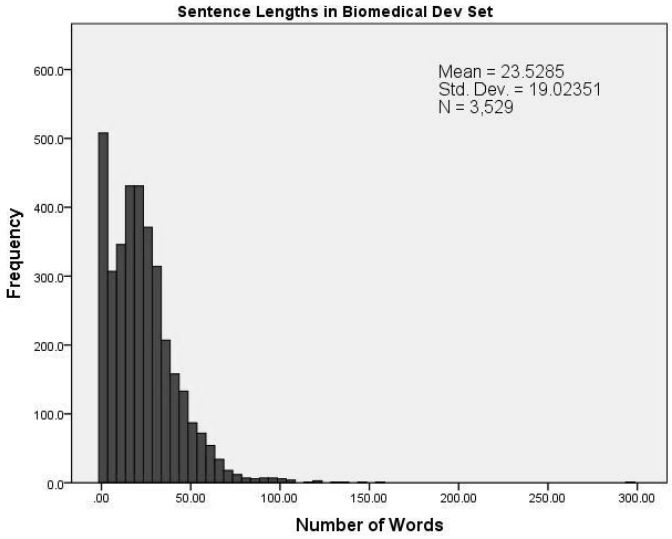
[5]The short sentences include headings, figure and table names, references to journals or legal authorities, and the like.



(a) DUC articles (max = 90 words)



(b) Legal cases (max = 261 words)



(c) Biomedical articles (max = 297 words)

Figure 2.1: Distribution of sentence lengths in DUC data is nearly normal. Distribution of sentence lengths in dev sets of legal cases and biomedical articles is noticeably skewed, with a large number of very short sentences and a long tail of very long sentences. Maximum sentence lengths for both were over 250 words.

each document. For biomedical articles, we limited summary length to 20% of the original document, following Reeve et al. (2007) and Ou et al. (2007). For legal cases, we limited summary length to the length of the human-generated summary for that case, which for this study were on average 26% of the length of the input cases.

2.2.2 Existing Systems

To solve the problem of summarizing documents that include long, complicated sentences, the current work tries to make the sentences shorter before running a summarization algorithm. We compare three such methods: an existing compression method, an existing simplification method, and our disaggregation method.

2.2.2.1 Compression System

For the compression system, we used Napoles's implementation[6] of the (Clarke and Lapata, 2008) Integer Linear Programming (ILP) algorithm for sentence compression. A complete explanation of ILP is beyond the scope of this paper. Briefly, though, ILP seeks to identify the values for decision variables that maximize a linear objective function.

In this case, the objective function combines a language model with a significance scoring function. The language model, trained on the English Gigaword corpus (Graff and Cieri, 2003), allows it to estimate the probability that each unigram, bigram, and trigram in a proposed compression would occur in English. Thus, for example, given the sentence

The dogs barked at the hissing cats. (7)

the language model portion of the objective function would tend to prefer the compression

Dogs barked at cats. (8)

over

[6]https://github.com/cnap/sentence-compression

The barked the hissing. (9)

The significance scoring portion of the function assigns words a weight representing their significance, to ensure that the compression contains the most important words of the original sentence.

The system also includes constraints. For example, a modifier will not be included in the compression unless its headword is included. This prevents a compression that includes "hissing" but excludes "cats."

ILP-based compression for summarization generates improved output when the ILP performs both compression and sentence selection jointly (Chali and Hasan, 2012), but here we have taken a pipeline approach of compressing and then selecting sentences. The pipeline is necessary to allow us to compare apples to apples: if we used a joint compression and selection system, we would not be able to determine whether differences in performance between that and the other systems was due to the method of shortening sentences or due to the difference between sentence selection algorithms.

2.2.2.2 Simplification System

For sentence simplification, we use the hybrid rule-based and learned simplification model of Siddharthan and Mandya (2014). The system combines handwritten rules for syntactic simplification and rules learned by translation system trained on aligned English Wikipedia and Simple English Wikipedia texts. The system recognizes patterns in dependency parses of sentences, applies rewrite rules to the parse tree according to which pattern(s) it matched, and generates the simplified sentence text from the altered parse tree. For example, the paper describes a handwritten syntactic simplification rule to replace passive with active voice, shown in Figure 2.2. Given a sentence

The cats were chased by a dog. (10)

RULE: PASSIVE2ACTIVE
1. DELETE
(a) nsubjpass(?X0, ?X1)
(b) auxpass(?X0, ?X2)
(c) agent(?X0, ?X3)
2. INSERT
(a) nsubj(?X0, ?X3)
(b) dobj(?X0, ?X1)
3. NODE OPERATIONS
(a) AGR-TENSE:?X0←?X2
(b) AGR-NUMBER:?X0←?X3

Figure 2.2: A rule from the simplification system for replacing passive voice with active voice.

(modified from an example in the paper) with a dependency parse as shown in Figure 2.3a, this rule would generate the new parse tree shown in Figure 2.3b, which in turn would generate the simplified sentence

A dog chased the cats. (11)

det(cats-2, The-1)
nsubjpass(chased-4, cats-2)
auxpass(chased-4, were-3)
root(ROOT-0, chased-4)
det(dog-7, a-6)
agent(chased-4, dog-7)

(a) Original Parse

det(cats-2, The-1)
root(ROOT-0, chased-4)
det(dog-7, a-6)
nsubj(chased-4, dog-7)
dobj(chased-4, cats-2)

(b) Simplified Parse

Figure 2.3: The dependency parse of model sentence 10 before and after applying the simplification rule in Figure 2.2.

The automatically generated rules work similarly but focus on lexical and lexico-syntactic simplification. Thus, for example, there is a rule that, iff X_1 is a word from the set "[extensive, large, massive, sizable, major, powerful, unprecedented, developed, giant]," will

delete amod(X_0, X_1) and replace it with amod(X_0, X_2), where X_2 is "big." The rule will also transfer any subtree rooted at X_1 to instead have X_2 as its root.

The simplifier permits five operations:

1. Delete: remove the specified dependency relations from the old sentence
2. Insert: add the specified dependency relations to the new sentence(s)
3. Order: process the daughters of a node of the dependency graph in the specified order
4. Move: delete the specified node (as opposed to just the relation) from the graph, and attach its children to a specified other node or, if unspecified, to the parent of the deleted node
5. Node Operations: make morphological changes to a specified word to ensure its agreement with other words in the sentence

The explicit goal of this system is to make sentences easier to read, not simplification for summarization. It is nevertheless well suited for comparison with the other systems, not only because it is a state-of-the-art simplification system, but also because it has performed particularly well on tests of meaning preservation, perhaps because it does not delete information through sentence compression. It therefore can be expected to be particularly competitive with the disaggregation system in generating sentences that are shorter than the source sentence but still retain all of the meaning.

2.2.3 Disaggregation System

We built the sentence disaggregator from a modified version of the (Siddharthan and Mandya, 2014) sentence simplifier. Our changes fell into two categories: changes to rules and changes to the system that implements the rules.

2.2.3.1 Modification of the Simplification System for Disaggregation

In addition to the five operations of the simplification system described above, we found a sixth operation, copy, necessary for disaggregation for summarization, due to a difference in the best way to handle duplicate subtrees. Sometimes when a sentence is split, the same subject must be used in both sentences. For example, in the sentence

The blue and yellow balloons floated above the ground and drifted in the breeze. (12)

"balloons" is the subject of both "floated" and "drifted." If the resulting split sentences will be read together, it is redundant to say

The blue and yellow balloons floated above the ground.
The blue and yellow balloons drifted in the breeze. (13)

and would instead be more desirable to simplify to

The blue and yellow balloons floated above the ground.
These balloons drifted in the breeze. (14)

The simplifier therefore replaces the subtree that depends from this repeated subject with the determiner "this" or "these" as appropriate.

In a summary, though, where one of the resulting sentences might be extracted without the other, this substitution could cause confusion or inaccuracy. For example, the summary could misleadingly say

The red balloons had not yet been inflated. These balloons drifted in the breeze. (15)

A better disaggregation for extractive summarization requires including the subtree that descends from "balloons" instead of replacing it with a determiner. In other circumstances, though, the flexibility to exclude some or all of the subtree may be desirable. For instance, it is preferable to split

The court held that community antenna television ("CATV") systems, which retransmitted signals, did not infringe. (16)

into

The court held that community antenna television ("CATV") systems did not infringe. CATV systems retransmitted signals. (17)

Disaggregation for summarization therefore benefits from the option to include the subtree or not. Hence we modified the simplifier so that rules could specify that a word should be copied without its subtree.

2.2.3.2 Disaggregation Rules

With the modified system in place, we identified disaggregation rules. First, we removed all rules from the original system that involved lexical substitution, as well as rules for converting passive to active voice and rules standardizing quotations. These are pure simplification rules and are not a part of disaggregation. We retained most rules for splitting appositions, relative clauses, subordination, and conjunction into separate sentences. Some of these rules were not well suited for summarization, though. For example, the simplifier included a rule that splits a sentence such as

The court reduced the term of imprisonment after considering the factors set forth in section 3553(a). (18)

into

The court reduced the term of imprisonment.
This happened after considering the factors set forth in section 3553 (a). (19)

If the second sentence is included in a summary without the first, the reader will be left wondering *what* happened after these factors were considered. We therefore disabled 21 such rules, leaving 94 of the original, handwritten syntactic simplification rules in place.

Preliminary testing showed that sentences in legal cases and biomedical articles often could not be split by these rules. We therefore wrote additional disaggregation rules

designed for these complex sentence structures. But choosing what rules to write is a subjective task. Therefore, to choose the best disaggregation, one of the authors prepared a preliminary set of disaggregation instructions, and then the authors and a colleague from the same lab individually disaggregated a set of ten sample sentences from the development set using those instructions. The resulting disaggregations were compared. The researchers discussed the best disaggregation of each sample sentence for the summarization task. When consensus was reached, the result became the gold standard. Manual rules were then written based on the gold standard models and variations on them.

For example, one original sentence in the gold standard was

> *Just before the passage of the 1976 Copyright Act, the Supreme Court held in Fortnightly Corp. v. United Artists Television, Inc. and Teleprompter Corp. v. Columbia Broadcasting System, Inc. that community antenna television ("CATV") systems—which captured live television broadcasts with antennas set on hills and retransmitted the signals to viewers unable to receive the original signals—did not infringe the public performance right because they were not "performing" the copyrighted work.* (20)

The gold standard called for the relative clause set off by em-dashes (—) to be split off and become two new sentences:

> *CATV systems captured live television broadcasts with antennas set on hills.*
> *CATV systems retransmitted the signals to viewers unable to receive the original signals.* (21)

Although the underlying simplification system included rules to separate relative clauses into their own sentences, such rules required the dependency parser to use the "rcmod" dependency. The parser did not apply this relationship to relative clauses set off by em-dashes rather than commas. We therefore needed a rule to recognize the actual dependency parse of a relative clause without relying on the rcmod relationship to signal it. For this gold standard sentence, the rule would need to make "systems" the subject of the new sentences and keep the "CATV" abbreviation modifying it. Such a rule was therefore specific to sentences with a token in an abbreviation relationship with the noun that would become the new subject. But of course, we wanted the system to recognize relative clauses without an

abbreviation, and this required a related rule.

In total, our disaggregation system applied 112 rules—94 from the simplifier, plus 18 that we developed using this approach. The difference in output between our disaggregator and the original simplifier is illustrated in Table 2.2.

Original Sentence	Disaggregated Sentence	Simplified Sentence
Actinic keratosis, also known as senile keratosis, results from the proliferation of atypical keratinocytes as a consequence of long exposition to ultraviolet radiation and it has been considered a premalignant lesion which may evove[sic] to squamous cell carcinoma.	Actinic keratosis are also known as senile keratosis. Actinic keratosis results from the proliferation of atypical keratinocytes as a consequence of long exposition to ultraviolet radiation. It has been considered a premalignant lesion. This premalignant lesion may evove[sic] to squamous cell carcinoma.	Actinic keratosis, sometimes called as senile keratosis , results from the proliferation of atypical keratinocytes because of long exposition to ultraviolet radiation and it has been thought a premalignant lesion. This lesion may evove[sic] to squamous cell carcinoma.

Table 2.2: Examples of the difference in output of the disaggregation system and the simplifier

2.2.4 Summarization System

For sentence selection, we use the graph-based LexRank summarizer (Erkan and Radev, 2004b). In the LexRank Only condition of our experiment, we run LexRank on the sentences of the original document to sort its sentences in descending order of importance. In the Simplified, Disaggregated, and Compressed conditions, we instead use the simplified, disaggregated, or compressed sentences as input to LexRank.

LexRank represents sentences as nodes of a graph. The graph of the document includes an edge between a pair of sentences if their cosine similarity exceeds a certain threshold. Edges coming into a sentence indicate that it is important to the document, so sentences "vote" for related sentences. However, a measure based purely on the degree of a node would allow a group of outlier sentences that were closely related to each other, but not to

the main thrust of the document, to create an illusion of centrality. To avoid this problem, LexRank weights a node's votes using the node's centrality.

Starting with the most important sentence, we select sentences to add to the summary until we encounter one that would cause our summary to exceed the maximum word count. But because we are dealing with sophisticated documents containing long sentences, we may have found a particularly long sentence that could not be split or compressed, and there may in fact still be room in the summary for one or more normal-length sentences. We therefore check at this point to see if one of the next three most important sentences could be added to the summary without exceeding the word count. If so, we add it and continue adding the sentences after it, following the same rule when we encounter another sentence that would make the summary exceed the maximum word count. But if none of the next three most important sentences is short enough to be added, we stop adding sentences to the summary, to avoid adding a collection of unimportant but very short sentences simply because they fit.

Once the system has identified the sentences to include in the summary, it puts them in the order in which they appeared in the original document.

2.3 Experimental Evaluation

2.3.1 Evaluation Methodology

Human-judged evaluation is necessary, since automatic evaluation using ROUGE would not be appropriate for this experiment. ROUGE uses n-gram overlap between the summary and model summaries. That would bias the evaluation against the simplification system, which substitutes simpler words for more complex ones. In addition, automatic evaluations have not been shown to correlate with human evaluations of summaries when the sentences included in the summary are not precise cut-and-paste extractions from the original document. Thus, we use two types of human evaluation.

The first is an intrinsic evaluation, where evaluators rate each summary on the DUC quality questions.[7] As noted by Ou et al. (2007), human participants can judge the a summary directly, rather than compare it to an "ideal" summary.

The second is an extrinsic evaluation. As Daumé and Marcu (2005) argue, extrinsic evaluation has an advantage over intrinsic evaluation: it does not merely determine how similar the output is to the way a human would do it, but determines how useful the output is for the desired task. If we wish to generate summaries that convey important information from a document, we should test whether someone who has read the summary has absorbed that information. We therefore followed Otterbacher et al. (2008)'s approach, developing comprehension questions and comparing evaluators' ability to answer those questions using summaries output by the different pipelines.

Specifically, we randomly selected three cases and three articles from the test set of the corpus described above.[8] For each document, one researcher wrote five comprehension questions. Each question is multiple choice and can be answered based on the full text of the original document with no outside knowledge. Example questions appear in Table 2.3. Each evaluator was asked to answer the questions regarding the cases and articles.

For each biomedical article, the evaluator was assigned to one of four experimental treatments, which determined what form of summary would be given to them:

1. "LexRank Only:" Summary of source document with no simplification, disaggregation, or compression of sentences

2. "Simplified:" Summary of document after sentence simplification

3. "Disaggregated:" Summary of document after sentence disaggregation

4. "Compressed:" Summary of document after sentence compression

[7] http://duc.nist.gov/duc2004/quality.questions.txt

[8] Since humans were evaluating the summaries, we began with a smaller number of summaries than we could have evaluated with ROUGE. However, as discussed in Results and Discussion, *infra*, clear patterns emerged even with only six documents, so it was unnecessary to test using a larger data set.

What part of the lower court's decision does Rodriguez-Ocampo appeal? a. Denial of his motion to suppress statements made before he was advised to his right to counsel b. His conviction of two counts of illegal entry for a single offense c. Allowance of the prosecution's motion to admit portions of his juvenile record d. A sixteen-level sentencing enhancement e. I can't answer this question using this summary.
What method did the researchers use to collect data? a. Review of medical records b. Pre- and post-operative interviews with patients and their families c. fMRIs 3, 6, 9, and 12 months post-surgery, and annually thereafter d. Self-report using a smart-phone app e. I can't answer this question using this summary.

Table 2.3: Examples of multiple choice comprehension questions used for extrinsic evaluation

For legal cases, evaluators could be assigned to any of treatments one through four, or a fifth condition:

5. "Human:" Human-generated summary.

Fourteen evaluators participated in the experiment. All were college graduates. Three reported at least one year of law school. Two reported studying biology or medicine at the college level or above for a year or more.

Each evaluator was randomly assigned to one condition per original document. We then adjusted the assignments slightly so that each condition/document pairing received approximately equal numbers of evaluations (either three or four). Finally, we ensured that no evaluator saw the same document in more than one condition and that no evaluator saw the same condition for more than two documents.

Evaluators were told that they would answer "opinion questions" asking about their impressions of the quality of the summary and "information questions" to help the researchers "understand how well the summary conveys information from the original document." The instructions acknowledged that summaries could contain grammatical errors or be missing information. Since the goal of this extrinsic evaluation was to determine how useful output

summaries are, and summaries that fail to convey information—either because they do not contain it or because they are so ungrammatical that they are too difficult to read—are unlikely to be useful, the instructions pointed out the availability of option E for questions the evaluator could not answer. Evaluators were asked not to consult outside sources to answer the questions.

As noted above, the word count upper bound for biomedical article summaries was 20% of the word count of the original document, and for legal cases it was the word count in the corresponding human-written summary. Actual word counts for the six test documents are shown in Table 2.4.

Original Document	Summaries (Mean)
2011	611
4110	632
9013	1701
7156	1416
4745	843
5594	1076

Table 2.4: Word count in the original document and mean word count across summaries of that document.

2.3.2 Results

For each summary evaluation, we calculated a single comprehension score, which reflected the number of comprehension questions about that summary that an evaluator answered correctly. Since we asked five comprehension questions for each summary, these scores could range from zero to five. Mean comprehension scores appear in Table 2.5 and Figure 2.4.

As expected, scores on comprehension questions were best when evaluators used human-written summaries. Contrary to our expectation, however, the next highest performances were those in the LexRank Only group, followed by the Simplification group, with the Disaggregation and Compression groups lagging behind. A one-way ANOVA using Tukey's

Condition	Mean
Simplified	3.11
Compressed	2.00
Disaggregated	2.26
LexRank Only	3.50
Human	3.75

Table 2.5: Mean comprehension scores for each condition.

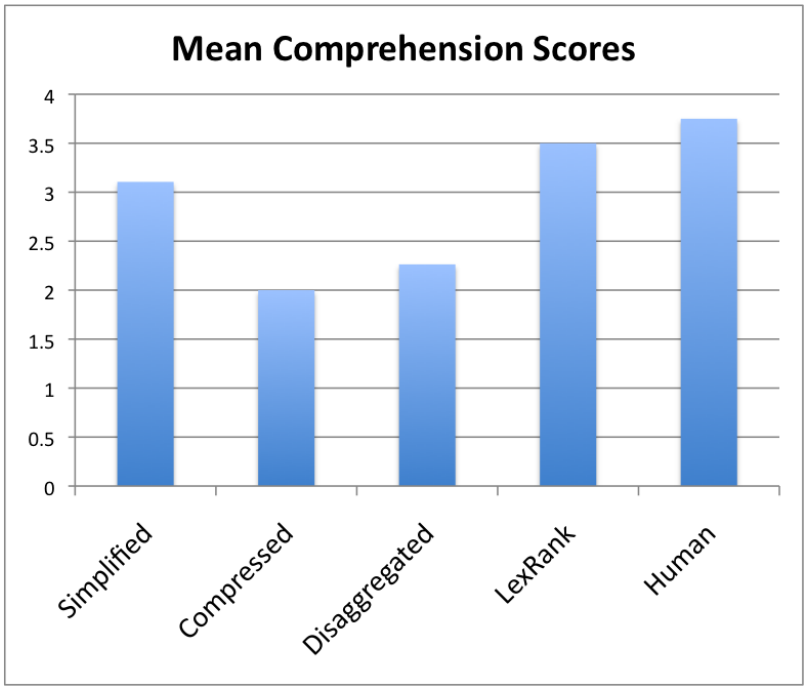


Figure 2.4: A comparison of mean comprehension scores achieved by the different systems.

HSD revealed that the differences between LexRank and Compression, LexRank and Disaggregation, and Human and Compression were significant at the $p = .05$ level; the other differences were not significant. A two-way ANOVA to determine whether the document type (legal case versus biomedical article) interacted with condition revealed no significant effects.

Because the quality questions seek information about different aspects of summary quality, we could not combine the seven quality questions as we could the comprehension questions. Thus, each summary evaluation received seven different quality scores, Q.1 through Q.7. Each score ranged from 4 points for "a," the most positive evaluation, to 0 for

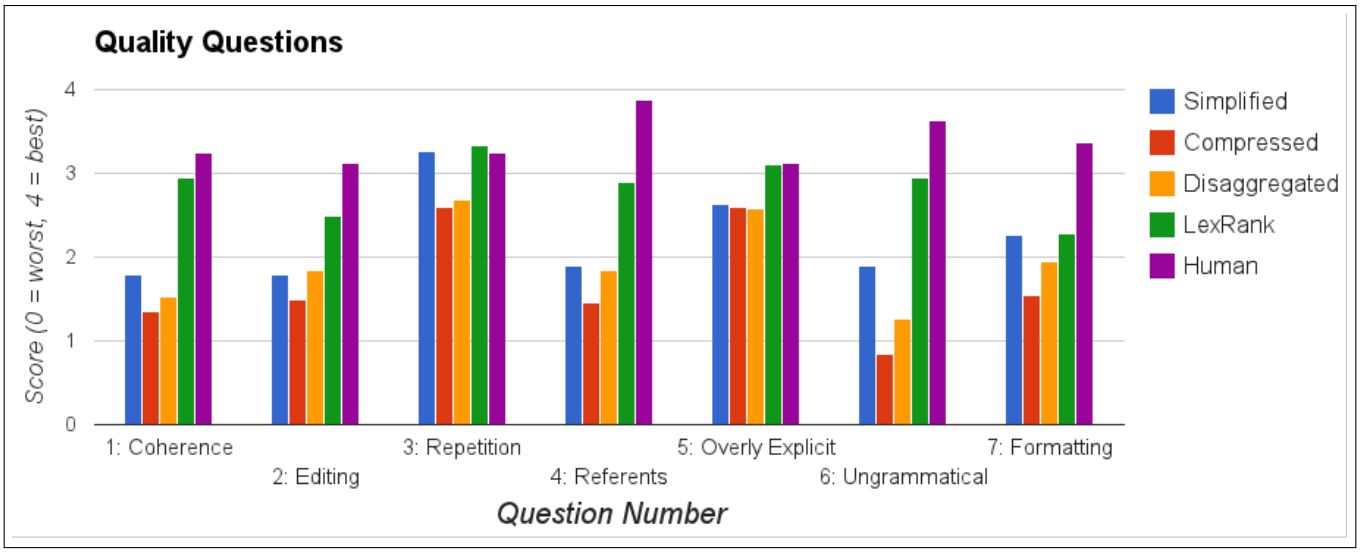


Figure 2.5: A comparison of scores of the different systems on the DUC quality questions.

"e," the most negative evaluation.

Performance of the different systems on the quality questions is summarized in Figure 2.5. Although the ratios between the systems differ, the pattern for all questions except for 3 and 5 is quite similar to the pattern seen in Figure 2.4. As expected, the human-generated summaries usually scored best on the quality questions. On questions 3 and 5, the LexRank Only summaries performed as well as or better than the human-generated summaries; however, the differences were not significant. Generally, summaries from the Simplified condition performed slightly worse on the quality questions than the LexRank Only summaries, followed by Disaggregated, with Compression ranked last. Significance of differences varied. On questions 3 and 5, there were no significant differences between any of the conditions. LexRank Only and Human were never significantly different. Means and significant differences for all seven quality questions in all five conditions are available online at http://www-personal.umich.edu/~cfdollak/sophisticated_documents_paper/quality_questions_appendix.pdf.

Except for questions 3 and 7, all possible pairings of the quality question scores showed a significant positive correlation using Pearson's r. Scores on information questions were significantly positively correlated with scores on all seven quality questions; these correla-

tions appear in Table 2.6.

Quality Question	1	2	3	4	5	6	7
Pearson's r	0.410	0.438	0.288	0.418	0.330	0.440	0.286

Table 2.6: Pearson's r correlation coefficients between scores on comprehension questions and quality questions. All are significant at the .01 level or better.

2.4 Discussion

2.4.1 Analysis of Results on Quality Questions

With the exception of question 3, the results on the quality questions followed a consistent pattern of Human/LexRank performing best, followed by Simplification, Disaggregation, and Compression, in that order. It is not surprising that the human-written summaries perform well on this portion of the evaluation; if anything it is surprising that they did not perform even better than they did.

The lack of a significant difference between LexRank and Human conditions on any of the quality evaluations appears impressive, although we must be cautious not to read too much into it. Because the Human condition applied only to the legal cases, not to the biomedical articles, the sample size of the Human group was smaller than those of the other conditions ($N_{Human} = 8; N_{LexRank} = 18$). A larger study might find significant results where this one did not. In addition, it is possible that by showing evaluators summaries from the much lower quality Simplified, Compressed, or Disaggregated conditions alongside LexRank and Human summaries, we may have distorted their perception of the quality scale. A truly fair test of LexRank's performance would require a larger number of summaries and only the Human and LexRank conditions.

We can say with confidence, though, that the LexRank summaries outperformed the Compressed and Disaggregated ones on all of the quality questions and the Simplified system on most of them. On some quality questions, this is unsurprising. The extra step of

simplifying, compressing, or disaggregating a sentence adds an opportunity for a previously grammatical sentence to become ungrammatical. In addition, removing part of a sentence increases the likelihood of having unclear referents.

However, our hypothesis suggested that on quality question 2, which asks how much useless or confusing material ought to be removed from the summary, we could expect Simplified, Compressed, and Disaggregated conditions to perform well. After all, their purpose was to automatically remove useless material. Yet all three performed significantly worse on this question than the Human condition, and Compressed also performed significantly worse than LexRank.

One possible explanation is that the compound question has blurred the issue. Perhaps the evaluators felt that they would need to edit a great deal of confusing material from the Simplified, Compressed, and Disaggregated summaries, but were neutral or even positive as to the amount of useless material. Another possibility is that the experimental systems rendered some sentences so confusing as to be useless.

A review of the actual summaries supports this second view. Table 2.7 contains the first hundred words from the LexRank, Simplified, Compressed, and Disaggregated versions of the same summary.[9] In the Compressed and Disaggregated summaries, and to a lesser extent the Simplified summary, sentences have been altered and taken out of context in ways that make them quite confusing. For instance, in the Simplified summary, the heading "Austin Bank—Troup, Texas," which in the original document introduced a recitation of the facts regarding the robbery of that bank, was transformed into the rather bewildering sentence "Austin Bank is Troup, Texas." This type of change explains why, even if some useless material was excluded by the Simplified, Compressed, and Disaggregated conditions, evaluators would still score these conditions as needing to have more useless or confusing material trimmed.

Additionally, we note that question 3, which asks whether the summary is repetitive,

[9]The summaries in their entirety are available online at http://www-personal.umich.edu/~cfdollak/sophisticated_documents_paper/example_summaries.pdf.

a: (2010) UNITED STATES of America, Plaintiff-Appellee, v. Paul Edward THOMAS; Derrick Van Hodges, Defendants-Appellants. Each robbery was completed within two minutes. The basis for the warrant was DNA evidence linking Hodges to a glove dropped during a bank robbery in Henderson, Texas. Thomas and Hodges were named in an 18-count indictment charging them with conspiracy, bank robbery, and weapons offenses related to the following bank robberies: 1. DISCUSSION I. Sufficiency of the Evidence Thomas and Hodges argue the government presented insufficient evidence identifying them as the bank robbers. Thomas claims the evidence against Hodges was much stronger and implies that Thomas was found guilty by association.
b: Each robbery was completed within two minutes. Thomas and Hodges were named in an 18-count indictment charging them with conspiracy , bank robbery , and weapons offenses related to the following bank robberies : 1. Austin Bank is Troup, Texas. DISCUSSION I. Sufficiency of the proof Thomas and Hodges argue the government presented insufficient evidence identifying them as the bank robbers. Thomas claims and the proof against Hodges implies that association found Thomas guilty. The proof was much stronger. Hodges was ‘ weak in some instances. This happened after a bank robbery was also ‘ happenstance. United States v. Clayton . We apply this standard of review to direct and circumstantial evidence.
c: (2010) UNITED STATES of America, Plaintiff-Appellee, v. Paul Edward THOMAS; Derrick Van Hodges, Defendants-Appellants. Ireland, Carroll , Kelley for Thomas is P.C. Ireland, Carroll , Kelley for Thomas is Tyler. Ireland, Carroll , Kelley for Thomas is TX. Hodges had in his possession a $ 10 bait bill taken a week earlier during the robbery of a bank in Crockett, Texas. At that time, arrested. America Henderson, is Texas ; 3. Austin Bank is Troup, Texas. Thomas, Hodges , were jointly tried before a jury. Thomas claims.the evidence against Hodges implies that Thomas was found guilty by association. The evidence against Hodges was much stronger.
d: LESLIE SOUTHWICK Circuit Judge Paul Edward Thomas and Derrick Van Hodges were convicted of counts of conspiracy bank robbery and weapons possession . Bank robberies were executed in manner . Each robbery was completed within two minutes . Derrick Van Hodges was arrested in Tyler Texas on state warrant . The basis for the warrant was evidence linking Hodges to glove dropped during a bank robbery in Henderson Texas . Bait bill was found in child ’s bedroom . DISCUSSION Sufficiency of Evidence Thomas and Hodges argue government presented evidence identifying them as bank robbers . Thomas claims the evidence against Hodges was stronger and implies that Thomas was found guilty by association .

Table 2.7: Segments of summaries output by (a) LexRank Only, (b) Simplified, (c) Disaggregated, and (d) Compressed conditions.

is the only quality question that does not follow the pattern of the others, as can be seen in Figure 2.5. For question 3 only, the Simplification system performed approximately as well as the LexRank Only and Human conditions. Because the differences in question 3 were not statistically significant, this apparent deviation from the pattern may be illusory. But this result could also suggest that the simplifier's use of determiners instead of repeated noun phrases helped avoid overly repetitive summaries.

Another notable result was that Simplified and Disaggregated were not significantly different on any quality measures except for question 6, which asks about ungrammatical sentences. Given the engineering described above to avoid introducing ambiguous determiners and to repeat modifiers in the split sentences, we had expected that the Disaggregated system would perform better on question 4, which asks how difficult it was to identify referents of noun phrases.

An important question is why Disaggregated summaries included more ungrammatical sentences than Simplified summaries. The most likely cause seems to be overfitting. The modifications and the additional rules described above were based on a small number of gold standard sentences. Changes to the system intended only to allow modifiers to be repeated generated some side effects in initial testing; for example, some disaggregated sentences contained long strings of repeated conjunctions. These problems were fixed, in that they no longer occurred when tested on the gold standard sentences and basic modifications to them, before the summaries in this study were generated. However, perhaps the gold standard sentences and modifications to them did not expose a broad enough range of possible problems, and other side effects remained that could only have been discovered by a larger system test. Similarly, the new rules were developed to work on the sentences from the gold standard collection and variations on those. It is possible broader testing might reveal sentences that match the dependency patterns we found in those sentences, but are grammatically different enough that the rule application no longer makes sense.

Perhaps the most surprising result on the quality questions was the poor performance

of the Compression system. The Clarke and Lapata ILP-based sentence compression algorithm that we used is widely considered state of the art in sentence compression. We suspect that the problem may relate to the language model that the algorithm incorporates in its objective function. Maybe the trigram model built from the LDC Gigaword corpus of newswire articles does a poor job of representing n-grams that show up in legal cases and biomedical articles. Biomedical articles in particular are likely to contain a great many out-of-vocabulary terms. A simple follow-up study could test this hypothesis by building a language model using a corpus of sophisticated documents and checking if performance improved. It would not require evaluating entire summaries, but could be evaluated on a sentence-by-sentence basis.

2.4.2 Analysis of Results on Comprehension Questions

The Compressed condition seems to have suffered from the predicted problem: the compression algorithm does not know what information will be important to the summary overall, and so it sometimes omits words that are actually needed to make a sentence meaningful. Given the paragraph

> *The mean time to loss of Engel Class II status after STL was 15.2 years (95% CI 13.2–17 years), and after mtg-SelAH it was 13.8 years (95% CI 11.9–16.2 years).*
> *The difference was not significant (p = 0.536).*
> *The mean time to loss of Engel Class I status after STL was 15.2 years (95% CI 13.2–17 years), and after mtg-SelAH it was 13.1 years (95% CI 11.9–16.2 years).*
> *The difference was not significant (p = 0.536).*
> *The mean time to loss of Class IA status after STL was 14.6 years (95% CI 12.2–17 years), and after mtg-SelAH it was 7.9 (95% CI 6.1–9.7 years).*
> *The difference was significant (p = 0.034) (Fig. 1).*

(22)

the compression was

The time to loss of Engel Class status was years years and it was years years .
The difference was not significant .
The time to loss of Engel Class I status was years years and it was years years .
The difference was not significant .
The time to loss of Class IA status was years years and it was 7.9 years .
The difference was . (23)

The original paragraph might have allowed the reader to infer that the answer to the comprehension question

Which of the following best describes the differences in seizure outcomes between the group that underwent standard temporal lobectomy (STL) and the group that underwent selective amygdalohippocampectomy (SelAH)? (24)

is most likely b,

b. The groups showed no difference in time to loss of Engel Class I or II status; STL performed better on time to loss of Engel Class IA status; and the SelAH group had more seizures during attempted medication withdrawal. (25)

since there was no significant difference between the groups on time to loss of Engel Class II or Engel Class I status, but there was a significant difference in time to loss of Class IA status. No such inference can be drawn from the compressed version of the paragraph, since important information is missing.

Since the Disaggregation condition generated the second-worst performance of all systems, though, the results on the comprehension questions do not support the hypothesis that the problem of compression removing important meaning will be solved by splitting, rather than compressing, the sentences. Instead, the similarity between the performance of Disaggregation and Compression suggests that they may suffer from similar problems.

There are two possible explanations for the poor performance of the Disaggregation and Compression systems on the comprehension measure. The first is that the information needed to answer the questions was simply not in the summaries, and the second is that the poor quality of the summaries obscured answers that were in fact present.

If the information needed to answer the questions was simply not in the summaries, we

need to understand why it was missing. The maximum word count for each summary was fixed across conditions, so all summaries had the opportunity to include the same amount of information. But the Disaggregation condition tended to generate repetitive sections of text. The LexRank algorithm has a weakness for repetition of a phrase in multiple sentences; it construes this as similarity between the sentences, and the sentences therefore "vote" for each other to be included in the summary. Thus, when faced with a case that listed counsel as

> *Laurel Franklin Coan, Jr., Asst. U.S. Atty. (argued), Robert James Middleton, Tyler, TX, for U.S.*
> *Deborah Johnson Race (argued), (Court-Appointed), Ireland, Carroll & Kelley, P.C., Tyler, TX, for Thomas.* (26)

the disaggregator mistakenly combined the fragments, then split them into

> *Laurel Franklin Coan, Jr. Asst.*
> *U.S. Atty.*
> *(argued), Robert James Middleton , Tyler TX for U.S. Deborah Johnson Race ( ) , (Court-Appointed) , Ireland , Carroll , Kelley for Thomas ,.*
> *Tyler TX for U.S. Deborah Johnson Race ( ), is argued.*
> *Ireland, Carroll , Kelley for Thomas is P.C.*
> *Ireland, Carroll , Kelley for Thomas is Tyler.*
> *Ireland, Carroll , Kelley for Thomas is TX.* (27)

And LexRank, noticing the repetition, included the last three lines in the summary. Flaws like this take up space that could be used to convey actually important information.

A second explanation is that, although the information is present, human readers could not extract it, because the generated summaries were so difficult to read and comprehend that they obscured the answers. Consider the comprehension question that was least often answered correctly:

What effect did surgery type have on psychiatric outcomes?
a. The STL group experienced increased depression
b. The STL group had increased paranoia, while the SelAH group had decreased paranoia
c. The SelAH group had increased paranoia, while the STL group had increased depression
d. The SelAH group had increased depression and anxiety
e. I can't answer this question using this summary.

(28)

The correct answer is b. The only evaluator who answered it correctly had a summary from the Compressed condition. The summary included the following relevant sentences:

Standard temporal was associated with higher scores on assessment of paranoia .
Our concern was that STL cause rates of de novo psychosis as have been associated with it .9,25,27 not a patient in group was diagnosed psychosis .
Gyrus SelAH be procedure for patients with high levels of disease paranoia .

(29)

In this case, the information needed to answer the question—that STL was associated with increased paranoia—was in the summary; however, an evaluator could easily misunderstand the second sentence to mean the researchers were concerned that STL would cause psychosis, but not a single patient actually exhibited psychosis.

The significant, positive correlation between comprehension score and all of the quality question scores lends some support to this last explanation: evaluators had more difficulty correctly answering comprehension questions as the subjective quality of the summary declined. Further study could ask annotators given a summary and a comprehension question with the correct answer to try to mark in the summary where the answer can be found, or indicate if the answer is not in the summary.

2.4.3 Qualitative Analysis of Shortened Sentences

While our quantitative results focus on the entire summarization pipeline, a brief qualitative examination of the sentence shortening methods on their own is enlightening.

The sentence compression method sometimes removed unneeded phrases, as in a compression that removed "On the other hand" from

On the other hand, argument of counsel is not evidence and is not to be considered as such by the jury. (30)

However, some compressions were ungrammatical; for instance

The DNA evidence and bait bills constitute sufficient evidence against Thomas to sustain convictions relating to the first and fifth bank robberies, and sufficient evidence against Hodges to sustain convictions relating to the first, second, fourth, and fifth bank robberies. (31)

was compressed to

The evidence and bait bills constitute evidence against Thomas sustain convictions relating to the first and bank robberies and evidence against Hodges sustain convictions relating to bank robberies. (32)

Additionally, the compressions often excluded words that were important to the meaning of the sentence; for instance,

Thomas and Hodges argue the government presented insufficient evidence identifying them as the bank robbers. (33)

was compressed to

Thomas and Hodges argue government presented evidence identifying them as bank robbers. (34)

Thus, as predicted, compression may substantially alter sentences' meaning.

Simplification, too, succeeded on some sentences; for instance, it neatly split

[W]e view the evidence and the inferences drawn therefrom in the light most favorable to the verdict, and we determine whether a rational jury could have found the defendant guilty beyond a reasonable doubt. (35)

into

(W) e view the proof and the inferences drawn therefrom in the light most favorable to the verdict.
And we determine whether a rational jury could have found the defendant guilty beyond a reasonable doubt. (36)

Yet the simplified sentences also suffer from some of the predicted problems. For example, sometimes lexical simplification changed the meaning, as when "challenging the lack of

eyewitness identification" was simplified to "challenging the rarity of eyewitness identification." In addition, simplification from passive to active voice sometimes obscured meaning; for instance, the simplification containing "association found Thomas guilty" could confuse a reader, while the disaggregation of the same sentence maintained the clearer statement that "Thomas was found guilty by association." And as predicted, the simplifier used "this" while the disaggregator kept entire noun phrases; for instance, where a simplification included

This eyewitness stepped outside of his office to observe traffic. (37)

the corresponding disaggregation included

Another eyewitness stepped outside of his office to observe traffic. (38)

as we hoped.

The disaggregation system still includes some simplification rules that should be changed for summarization. For instance, the original sentence

Still, if a joint trial would prejudice a defendant, district courts may sever the defendants' trials. (39)

should not have been disaggregated into

Suppose a joint trial would prejudice a defendant.
Then still district courts may sever the defendants' trials. (40)

Also, disaggregation occasionally caused some severe grammatical problems, particularly involving missing conjunctions.

2.4.4 Strategies for Improvement

Determining the cause of the difficulty in answering the comprehension questions helps determine the best way to improve future systems. If the problem is that valuable informa-

tion is being omitted from the summary because some characteristic of the disaggregated or compressed sentences causes suboptimal sentence selection, we might need to try using a different sentence selection algorithm. For instance, C-LexRank is an algorithm designed for multi-document summarization; it might handle the repetition that disaggregation adds to a document better than LexRank does.

If the problems are due to confusing, ungrammatical output of the disaggregator, however, then the focus of future work should be on improving disaggregation as a stand-alone function before doing further work on the summarization pipeline. Such future work could go in two different directions: improving the existing disaggregation method within the framework of the modified simplifier, or developing an entirely new method of disaggregation.

Certainly, there is room for improvement within the existing framework. Given that disaggregation introduced grammatical problems that simplification did not, it is apparent that testing of the disaggregation system on a broader collection of sentences is necessary to identify the source(s) of the problems.

However, there are several significant problems with the existing framework that make developing a new approach the better option. First, the disaggregator relies on a correct dependency parse to allow it to correctly split a sentence. However, we observed an unusually high rate of incorrect parses of sentences from the sophisticated documents; there were numerous problems with attachment, and some of the dependency relationships were not applied consistently across similar sentences. The problem may well arise because of the complexity of the sentences; the legal cases had a maximum parse tree depth of 47. Whatever the cause, the potential inaccuracy of the parser limits the ability to correctly disaggregate using this system.

Second, this framework cannot cope with a sentence structure that is very common to the longest sentences in legal cases: a numbered list within the sentence. For instance, our gold standard disaggregation transformed a sentence that began:

The request for a hearing must: (i) Provide a specific statement of the issue of law or fact to be raised or controverted ...; (ii) Provide a brief explanation of the basis for the contention; (iii) Demonstrate that the issue raised in the contention is within the scope of the proceeding;.... (41)

into a series of sentences:

The request for a hearing must provide a specific statement of the issue of law or fact to be raised or controverted.
The request for a hearing must provide a brief explanation of the basis for the contention.
The request for a hearing must demonstrate that the issue raised in the contention is within the scope of the proceeding.... (42)

The current framework has no way to iterate through list items. Thus, disaggregating a six-item list requires a rule for six-item lists, but that rule would not generalize to five-item lists. Writing rules for each number of items that might be on a list does not solve the problem, since each list item requires several variables, and the running time of the disaggregator becomes impractical when more than about ten variables are used. This system thus cannot effectively disaggregate list sentences, yet ignoring them leaves intact many of the longest sentences in legal cases.

Finally, the current framework is purely syntactic, but disaggregation is not a purely syntactic task. Consider the sentence

Rather than argue explicitly about the findings of the NRC, as to whether the portions of the contention met the reopening and/or the admissibility standards, in rejecting the Commonwealth's contention, Massachusetts devotes a substantial portion of its brief to arguing that the NRC acted arbitrarily and capriciously. (43)

Here, a purely syntactic approach cannot determine who rejected the Commonwealth's contention—the NRC or Massachusetts. Either semantic knowledge that "Massachusetts" and "the Commonwealth" refer to the same entity or some form of coreference resolution is necessary to know that a disaggregation containing

In rejecting the Commonwealth's contention, Massachusetts did not argue explicitly about the findings of the NRC. (44)

would not be accurate, while a disaggregation containing

> *Massachusetts does not argue explicitly about the findings of the NRC in rejecting the Commonwealth's contention.* (45)

would be.

For these reasons, future work should explore other ways of disaggregating sentences involving the retention of semantic coherence, rather than continuing to rely on the modified simplification framework.

2.5 Conclusion

We have found that sentence simplification, compression, and disaggregation before extractive summarization of sophisticated documents did not improve performance on extrinsic evaluations. The most likely reason is that, when applied to sentences from the legal and biomedical domains, all three sentence-shortening techniques produced some confusing or ungrammatical output. Future work should focus on improving the ability of sentence shortening techniques to handle sentences from these domains.

CHAPTER 3

Explorations of Paraphrase Detection

Two sentences are paraphrases of each other if both convey the same meaning. More formally, S1 and S2 are paraphrases if each entails the other; that is, knowing that S1 is true means that S2 must be true, and knowing S2 is true means that S1 must be true. Paraphrase detection is the task of determining whether a pair of input sentences is a paraphrase. The ability to identify paraphrases has numerous potential applications in natural language processing, such as summarization, translation, and question answering. However, paraphrase detection is not always straightforward, since the same information may be presented in very different ways, or different information may be presented in superficially similar ways. For instance, a human being can easily tell that

> Russia's military actions in Syria over the past several weeks were the subject of a meeting between Barack Obama and Vladimir Putin on Monday.

and

> At a meeting on Monday, the U.S. President and Russia's President Putin spoke about Russia's recent military buildup in Syria.

mean the same thing, even though they have only a few words in common, and the syntactic structures of the two sentences differ greatly. Similarly, humans are quite good at telling that

> John ate the fish.

and

The fish ate John.

mean very different things, even though they share all of the same words. The rest of this paper is organized as follows: In section 3.1, we discuss in more detail the previous work on paraphrase detection, as well as related work in deep learning. Section 3.2 describes our method. In section 3.3, we detail our experiments. We then present our results in section 3.4, and finally describe how future work might improve upon those results in section 3.5.

3.1 Related Work

3.1.1 Paraphrase Methods without Deep Learning

Early approaches to paraphrase detection emphasized lexical matching; for instance, Zhang and Patrick (2005) used rules to transform input sentences into canonical forms, then extracted lexical matching features. However, it quickly became apparent that simple lexical matching was not enough.

Mihalcea et al. (2006) described a semantic similarity method incorporating eight corpus- and knowledge-based measures of word-level semantic similarity, ranging from PMI to WordNet-based methods. The combination of these measures significantly outperformed simple lexical matching. Numerous others followed similar knowledge- and corpus-based approaches. Kozareva and Montoyo (2006) compared the performance of kNN, SVM, and MaxEnt classifiers; their feature set included word overlap, longest common subsequence, and word similarity features based on WordNet. Fernando and Stevenson (2008) used WordNet with a similarity matrix to recognize paraphrases that replaced words with synonyms or near synonyms. Ramage et al. (2009) performed a random walk over a graph that incorporated Wordnet and corpus statistics, with a bias towards the neighborhood of the bag of words representation of the input sentence. Islam and Inkpen (2009) measured

semantic similarity of sentences using the semantic similarity of the words (calculated with a Pointwise Mutual Information method) and modified versions of Longest Common Subsequence string matching. Ul-Qayyum and Altaf (2012) used LCS and bag of words to identify lexical overlap, then added a number of semantic heuristic features, such as synonymy and antonymy.

Others have emphasized the structure of sentences. Wan et al. (2006) used n-gram overlap, dependency relation overlap, dependency tree-edit distances, and difference in sentence lengths as features. Rus et al. (2008) combined lexical overlap enhanced with synonymy and antonymy information and dependency graph matching. Das and Smith (2009) combined a logistic regression model trained on lexical overlap features with a generative, quasi-synchronous grammar model that estimated whether the class paraphrase or not paraphrase maximized the probability of seeing the two sentences. Their approach thus took both lexical overlap and syntax into account. Heilman and Smith (2010) describe a tree-edit distance measure for semantic similarity; if the dependency tree of one sentence can be transformed into the other in relatively few steps, the sentences are more likely to be paraphrases. Bu et al. (2012) used a string rewriting kernel to measure the lexical and structural similarity between pairs of strings without having to construct syntactic trees. Bach et al. (2014) emphasizes that sentences are comprised of elementary discourse units and computes the similarity of sentences based on the similarities of their component discourse units.

Rather than attempting to detect sentence similarities, Qiu et al. (2006) sought to detect dissimilarities and determine if they were important. Their two-step process identified predicate argument tuples that could be aligned between the two sentences, then passed tuples that could not be aligned to a classifier that determined if the dissimilarity mattered.

Some latent variable models have been developed for this task in recent years. Guo and Diab (2012) use a latent variable model that builds a semantic profile of each sentence based on both the observed words and missing words of each sentence. Their weighted

matrix factorization approach is similar to SVD, except that it enables them to force the representation of missing words in a sentence to be zero. Xu et al. (2014) sought to detect tweets that are paraphrases using a joint word-sentence approach that assumes two tweets that share a topic and at least one "anchor" word pair are paraphrases. Their latent variable model is specific to the short context and unique wording that appears in tweets.

Several approaches have related paraphrase to machine translation. Wu (2005) used inversion-transduction grammars, which had previously been applied to machine translation and alignment, to outperform the baseline without using a thesaurus, lexical similarity model, or parameter training. Finch et al. (2005) used evaluation metrics developed for machine translation to predict whether two sentences were paraphrases. More recently, Madnani et al. (2012) was quite successful in training a meta-classifier using eight machine-translation metrics as features.

Vector and matrix based approaches have seen recent successes as well. Blacoe and Lapata (2012) compared shallow composition of word embeddings with deeper ones such as recurrent neural networks, and found that the shallow approaches performed approximately as well while requiring less computation. Milajevs et al. (2014) compared a number of types of word vectors using several compositional methods. For paraphrase, they found unlemmatized neural word embeddings gave superior results compared to co-occurence vectors.

Ji and Eisenstein (2013) invented a term-weighting scheme called TF-KLD, an alternative to the commonly used TF-IDF that takes advantage of paraphrase training data in calculating a term's weight. They built a matrix where each row represented an input sentence using this scheme, then applied a matrix decomposition technique called nonnegative matrix factorization to generate shorter vectors for each sentence. They compared sentence pairs using an SVM on the elementwise sum of the two sentence vectors concatenated with the absolute value of their elementwise difference. They also tested a simpler cosine similarity measure, but found the SVM more effective. Yin and Schütze (2015b) built on Ji

and Eisenstein (2013)'s method, modifying the TF-KLD scheme to TF-KLD-KNN, which handled words and phrases not seen in the training data using k-nearest-neighbors. They used embeddings not only for individual words, but also for both continuous and discontinuous phrases. Finally, they appended the eight machine translation metrics from Madnani et al. (2012) to the vectors they input into the SVM, generating a slight improvement in performance over previous methods.

3.1.2 Deep Learning

Hochreiter and Schmidhuber (1997) introduced the long short-term memory network (LSTM), and Gers et al. (2000) modified it to the form that is generally used today. Sutskever et al. (2014) describe a sequence to sequence learning structure that incorporates LSTMs.

Tai et al. (2015) introduced tree LSTMs for sentiment detection and semantic similarity; we describe their work in detail in Section 3.2.2. The current work is, to our knowledge, the first to apply their tree-LSTM to the problem of paraphrase detection. Moreover, we add an additional component to incorporate the labels of dependency types.

Several other deep learning architectures have been used for paraphrase detection with some success. Socher et al. (2011a) introduced unfolding recursive autoencoders for the problem. These learned to represent phrases using feature vectors. They then constructed a matrix that measured pairwise similarity between words and phrases of one sentence and words and phrases of another. Since this matrix could be variably sized, they used dynamic pooling to create a fixed-size matrix, to which they finally applied a classifier. Yin and Schütze (2015a) explicitly recognized the same point as Socher et al. (2011a): since it is not obvious which pieces of a pair of sentences might share the same meaning, it is valuable to try to align words and phrases at various levels of granularity. They therefore constructed a stacked convolutional neural network (CNN) with layers that generated a similarity matrix comparing the unigrams of each sentence, another comparing short n-grams, another comparing longer n-grams, and still another reflecting similarity for the

whole sentence. They then used dynamic pooling to convert these matrices into a single feature vector, to which they could apply logistic regression to classify the sentence pair as a paraphrase or not. He et al. (2015) also used a CNN for paraphrase detection.

Cheng and Kartsaklis (2015) compare the performance of LSTMs with recursive neural networks (RecNNs) for paraphrase. RecNNs are tree structured neural networks that have been used in several natural language processing (NLP) applications to compose vectors that represent the words of a sentence into a single vector by following the structure of the sentence's parse tree. (Irsoy and Cardie, 2014; Socher et al., 2013; Socher et al., 2011b) Cheng and Kartsaklis (2015) pretrain syntax-aware word embeddings and parameters for either an LSTM or a RecNN with a modification to do some word sense disambiguation. They then use the pretrained parameters in a *siamese architecture*. (Bromley et al., 1993) In the siamese architecture, two networks that share the same parameters generate a single vector representing each of the two sentences. The model then compares the two vectors using the L_2 norm or the cosine similarity and predicts whether the sentences are paraphrases. The present work differs from Cheng and Kartsaklis (2015) in several ways. First, we use tree LSTMs rather than either traditional LSTMs or RecNNs. Second, our tree LSTMs use the sentences' dependency parses, not the syntactic parses used to construct RecNNs in that work. Third, our model considers the dependency relation labels. And fourth, instead of cosine similarity or $L2$ norm, our model uses the method described in section 3.2.3 to determine, based on the vector representation of each sentence, whether the pair is a paraphrase or not.

3.2 Methods

3.2.1 Traditional LSTM

A long short-term memory network (LSTM) is a form of neural network that, like a recurrent neural network (RNN), is sequential: at each time step, it makes a prediction based on the input at that time step and the state of the network at the end of the previous time step. Unlike an RNN, however, an LSTM is suited to handle long-term dependencies. When faced with long sequences, RNNs often suffer from either a vanishing gradient problem or an exploding gradient. LSTMs introduce several gates that control the flow of information into and out of the "cell" at each time step. These gates have been shown to reduce the problem of vanishing gradients. (Hochreiter and Schmidhuber, 1997; Gers et al., 2000)

Mathematically, an LSTM can be described with the following formulas:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \tag{3.1}$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \tag{3.2}$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \tag{3.3}$$

$$u_t = \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}) \tag{3.4}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{3.5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.6}$$

x_t is the input at time step t; here, it is a word2vec embedding for the tth word of a sentence. h_0 and c_0 are initialized to zero vectors; subsequent h and c values are calculated according to equations (3.6) and (3.5). The W and U parameters are weight matrices, initialized to randomized orthogonal matrices and updated by subsequent training steps. The

b bias vectors are initialized to zeros and similarly updated. $\odot$ here represents element-wise multiplication.

Our first LSTM method inputs to this architecture a single string comprised of two sentences, separated by a delimiter, as a single sequence. For example we might input,

John ate the fish ::: The fish ate John

We then apply a final set of weights and bias to the h_n, where n is the length of the input sequence, and use a softmax to transform this output into a prediction of the class to which the sentences belong: paraphrase or not.

Our second sequential LSTM method uses a siamese architecture, as described in 3.2.3.

3.2.2 Tree LSTMs

We construct tree LSTMs for each sentence following the child-sum tree approach of Tai et al. (2015). As shown in figure , whereas an ordinary LSTM passes information sequentially through the sentence from the first word through the last, a tree LSTM passes information up the tree, so that the hidden vector h_j at node j incorporates the input vector x_j from that node, as well as the hidden vector h_k and memory cell c_k of each child k of j. Specifically, we calculate an intermediate $\tilde{h}$ that is the sum of the child h_ks:

$$\tilde{h}_j = \sum_{k \in C(j)} h_k \tag{3.7}$$

where $C(j)$ is the set of all children of node j. We use this intermediate value in place of h_{t-1} to calculate the i, o, and u values:

$$i_j = \sigma(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}) \tag{3.8}$$

$$o_j = \sigma(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}) \tag{3.9}$$

$$u_j = \tanh(W^{(u)}x_j + U^{(u)}\tilde{h}_j + b^{(u)}) \tag{3.10}$$

Since the purpose of a forget gate f is to control the flow of information from a child node's memory cell c_k, we need a forget gate for each child. Thus, we calculate

$$f_{jk} = \sigma(W^{(f)}x_j + U^{(f)}h_k + b^{(f)}) \tag{3.11}$$

We then apply each child's forget gate to its corresponding memory cell in calculating the value of the current memory cell, c_j:

$$c_j = i_c \odot u_c + \sum_{k \in C(j)} f_j k \odot c_k \tag{3.12}$$

Finally, h_j can be calculated from the output gate and the current memory cell as usual:

$$h_j = o_j \odot \tanh(c_j) \tag{3.13}$$

Note from equations (3.7), (3.11), and (3.12) that this architecture does not require knowing in advance the maximum number of children a node may have. This makes it well suited for use with dependency parse trees. For leaf nodes, we assume a single child with h_0 and c_0 set to zero vectors. Our input vector x_j is a word embedding for the word at node j.

Similar to Cheng and Kartsaklis (2015) and Yin and Schütze (2015a), we use a siamese architecture (Bromley et al., 1993). However, our network structure is based on a dependency parse of the sentence. We use clearnlp to obtain a dependency parse of each sentence. We then construct a child-sum tree LSTM based on that dependency parse.

Intuitively, one would expect the child-sum tree LSTM to outperform the LSTM for sentence pairs such as the one in figure 3.1, where the structure of the sentences affects

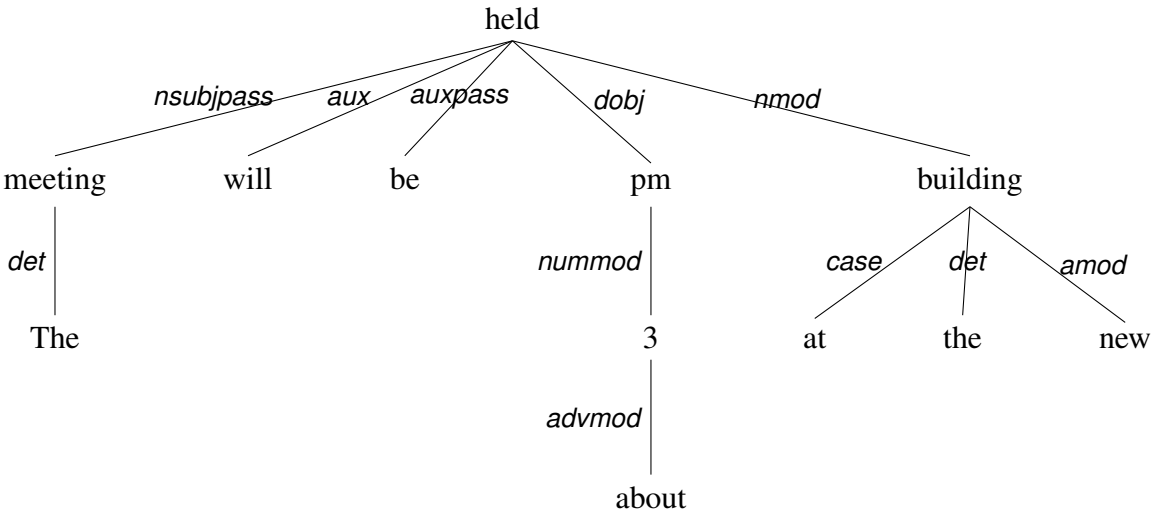
the similarity of their meanings. Moreover, the results of Cheng and Kartsaklis (2015) show that a tree-structured recursive neural network out-performs an LSTM at paraphrase recognition.

3.2.3 Siamese Architecture

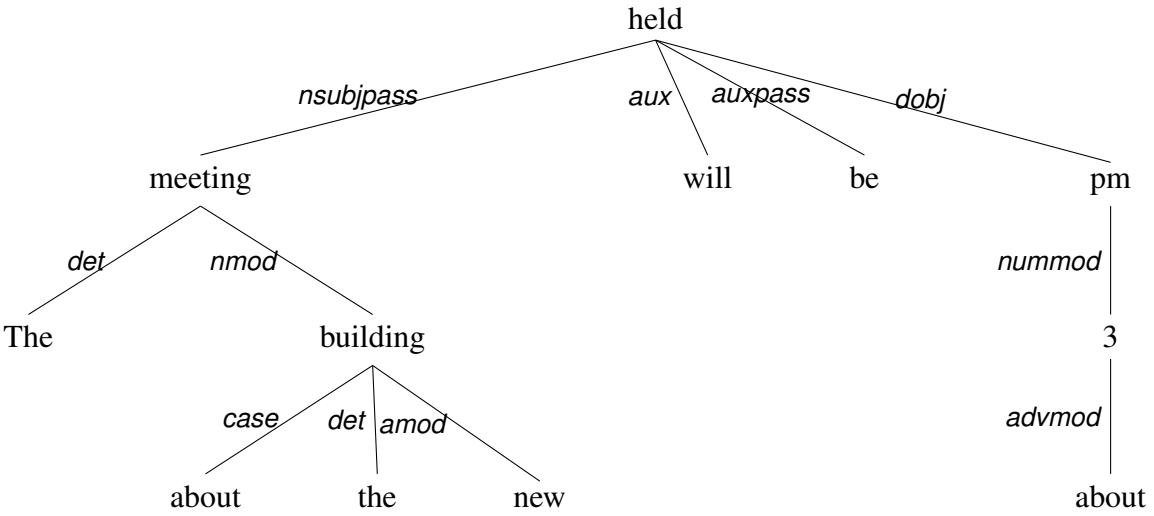
A Siamese architecture (Bromley et al., 1993) applies the network structure to two inputs in parallel. In the Siamese LSTM, we input two sentences, S_1 and S_2 into the LSTM. We apply the same parameters to each of them, generating a vector representation of each, h_{S1} and h_{S2}. We then create a vector h' that is the concatenation of $h_{S1} + h_{S2}$, $|h_{S1} - h_{S2}|$, and cosine similarity of h_{S1} and h_{S2}. This concatenation is in keeping with Ji and Eisenstein (2013), which found training an SVM on a concatenation of $h_{S1} + h_{S2}$ and $|h_{S1} - h_{S2}|$ more effective than setting a threshold cosine similarity, and Yin and Schütze (2015b), which appended cosine similarity to the vector described in Ji and Eisenstein (2013); however, neither of those works used the technique within a deep network. We apply a final set of weights W' and a final bias b' and take the softmax to generate our prediction. The method is essentially the same for LSTMs and tree LSTMs.

3.2.4 Pretraining

LSTMs tend to have a large number of parameters. Our LSTMs include weights for the x and h values at each timestep, as well as for the input, forget, and output gates. With this many parameters, a great deal of training data is essential. The effort that would be required for humans to label enough pairs of sentences as paraphrases or not would be herculean. We therefore pretrain the LSTM parameters using unlabeled data before training it on our more limited labeled data. Specifically, we modified our LSTM to act as a language model. At each time step, the goal of the model is to predict the next word of the input sentence. The cost function is the Euclidean distance from the predicted next word embedding to the



(a) "The meeting will be held about 3 pm at the new building."



(b) "The meeting about the new building will be held at 3 pm."

Figure 3.1: Although these two sentences contain identical words, their meanings are noticeably different. The dependency parse trees reflect that difference, and thus, child-sum tree LSTMs should have an advantage.

embedding of the actual next word. After pretraining, we import the pretrained parameter matrices into our LSTM classifier model and continue supervised training as usual. This is similar to the unsupervised pretraining technique in Yin and Schütze (2015a), but has a few distinctions. First, the Yin and Schütze technique was pretraining convolutional neural networks, not an LSTM. Second, their model was designed to predict the next word giving the three that preceded it, whereas here we use all of the sentence so far to predict the next word. Finally, they used noise contrastive estimation, whereas we use the Euclidean distance and

3.3 Experiments

Our experiments use the Microsoft Research Paraphrase corpus (Dolan et al., 2004; Dolan and Brockett, 2005). The corpus consists of 5801 sentence pairs, each hand-labeled as a paraphrase or not. We use the standard 4076/1725 training/test split.

Because LSTMs have so many parameters, a training set of fewer than 5000 examples is insufficient. We therefore follow Cheng and Kartsaklis (2015) and pretrain on examples from the PPDB paraphrase database (Ganitkevitch et al., 2013). PPDB is an automatically generated collection of lexical, phrasal, and syntactic identities and paraphrases. It includes 68.4 million phrasal paraphrase pairs—continuous strings of words—and 93.6 million syntactic paraphrase pairs—expressions including both words and non-terminals.[1] PPDB does not contain labeled negative examples, however. We generate negative examples by creating a similarity network of all phrases in the database and removing any edge that corresponds to a positive example, then using the highest-weighted edge for each node to create a pair.

To supplement our training data, we use the SICK data set (Marelli et al., 2014b). This corpus contains 9840 English sentence pairs from the ImageFlickr data set and the SemEval

[1] In addition, PPDB includes 4.9 million phrasal identities and 46.5 million syntactic identities, which we do not use.

2012 MSR-Video Description data set. Each pair was hand-labeled with an entailment relation between the two sentences (entailment, contradiction, or neutral) and a semantic relatedness score on a 5-point scale. To use SICK as training data for the paraphrase task, we gave each sentence pair a new binary label. Any pair that was labeled as entailment in SICK we labeled as a positive example; any pair that was a contradiction we labeled as a negative example; and for the remaining pairs, we used a cutoff relatedness score of 3.4. In this way we found 4920 positive pairs and 4920 negative pairs in the SICK data.

We implement our LSTMs and tree LSTMs using the Theano library. (Bastien et al., 2012).

3.3.1 Baselines

We consider several baselines. The simplest is a majority classifier: it determines whether there are more positive or negative examples are in the training set, and assigns the majority class to all test examples. The second baseline uses a bag-of-words (BoW) representation of each sentence, finds the cosine similarity between the pairs of BoW vectors, and trains a logistic regression or support vector machine (SVM) classifier that uses that cosine similarity as its sole feature. Our third baseline is similar to the BoW baseline; however, instead of a BoW vector, it represents each sentence by a vector that is the sum of the word embedding vector for each word of the sentence. We use pre-trained word embeddings from the GoogleNews-vectors-negative300 dataset (Mikolov et al. 2013). In some applications, a sum of these vectors would need to be normalized before being used, so that sentence length would not be a factor in vector size. Here, though, we are taking the cosine similarity between two vectors, so the lengths of the vectors does not matter, only their directions. Finally, we represent each sentence as a normalized sum of its component word vectors and adopt Ji and Eisenstein (2013)'s alternative to cosine similarity, which concatenates the sum of the two sentence vectors with the absolute value of their difference, then uses this as a feature vector for an SVM.

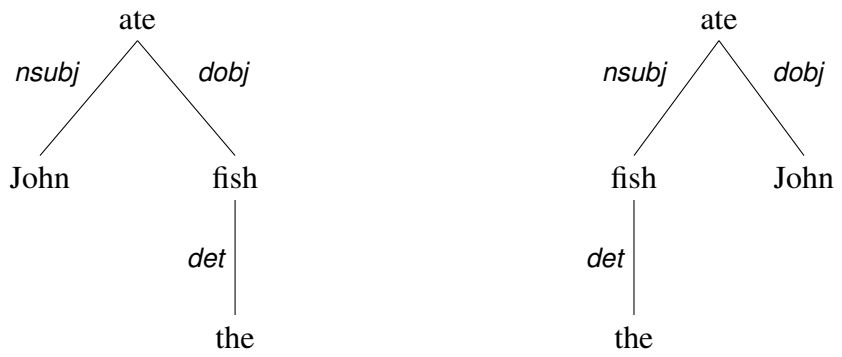
Method	Acc.	F1
Majority	68	??
Summed Embed Cos	70.4	81.1
Summed Embed Concat	70.2	81.8
Cosine BoW	72	82
Sequence LSTM	68.7	79.8
Sequence LSTM pretrained	68.9	80.0
Siamese LSTM	68.3	79.4
Siamese LSTM pretrained	68.8	78.3
Siamese LSTM with SICK	69.8	80.8
Siamese LSTM with pretraining and SICK	69.3	79.9
Siamese Tree LSTM	68.8	78.0
Socher et al. (2011a)	76.8	83.6
Yin and Schütze (2015a)	78.1	84.4
Ji and Eisenstein (2013) (training data only)	78.6	85.6
He et al. (2015)	78.6	84.7
Yin and Schütze (2015b)	78.7	84.8
Cheng and Kartsaklis (2015)	78.7	85.3
Ji and Eisenstein (2013) (with test data)	**80.4**	**85.9**

Table 3.1: Results on the MSRP corpus.

3.4 Results

Our results and those of other researchers are summarized in Table 3.1.

Ji and Eisenstein (2013)'s best reported results used "transductive learning," which included performing matrix factorization on both training data and unlabeled test data. Yin and Schütze (2015b) suggest that methods that do not use test data for training should be compared only with other methods that do not use test data for training. Here, we note Ji and Eisenstein's results both with and without test data.



(a) "John ate the fish." (b) "The fish ate John."

Figure 3.2: In a child-sum LSTM, $\tilde{h}$ being passed into "ate" in both of these sentences will be the sum of h representations of "John" and "the fish." Yet the dependency labels clearly indicate that these subtrees play different roles in the two sentences. Thus, a system that uses the dependency labels has an advantage here.

3.5 Potential Future Work

3.5.1 Label-Sensitive Tree LSTM

The example in figure 3.1 shows how a tree LSTM might perform better than an LSTM for certain sentences. However, consider the example sentences in figure 3.2. A simple child-sum tree will calculate the sum of a node representing "John" and a node representing "fish" with a dependent "the" for both sentences. Because sums are commutative, these will be the same. Thus, the child-sum tree LSTM would not have an advantage over the LSTM for that sentence pair. However, by looking at the dependency labels for the tree, it is clear that "John" plays a different role in (a) than in (b). We therefore might see improved performance by a new form of tree LSTM that *takes dependency labels into account*.

3.5.2 Additional Training Data

As noted above, an LSTM cannot reasonably be trained on 5000 example pairs; there are simply too many parameters. However, as Table 3.1 reveals, the pretraining data used in this study did not significantly improve performance. This may be because the SICK and

PPDB datasets were simply too different from the MSRP data; having access to a very large dataset that is similar to MSRP might improve performance.

CHAPTER 4

Effects of Text Corpus Properties on Short Text Clustering Performance

Corpora of collective discourse—texts generated by multiple authors in response to the same stimulus—have varying properties depending on the stimulus and goals of the authors. For instance, when multiple puzzle-composers write crossword puzzle clues for the same word, they will try to write creative, unique clues to make the puzzle interesting and challenging; clues for "star" could be "Paparazzi's target" or "Sky light." In contrast, people writing a descriptive caption for a photograph can adopt a less creative style. Corpora may also differ on how similar texts within a particular class are to one another, compared to how similar they are to texts from other classes. For example, entries in a cartoon captioning contest that all relate to the same cartoon may vary widely in subject, while crossword clues for the same word would likely be more tightly clustered.

This paper studies how such text properties affect the best method of clustering short texts. Choosing how to cluster texts involves two major decisions: choosing a similarity metric to determine which texts are alike, and choosing a clustering method to group those texts. We hypothesize that creativity may drive authors to express the same concept in a wide variety of ways, leading to data that can benefit from different similarity metrics than less creative texts. At the same time, we hypothesize that tightly clustered datasets can be clustered by powerful graph-based methods such as Markov Clustering (MCL) and Louvain, which may fail on more loosely clustered data. This paper explores the interaction

of these effects.

Recently, distributional semantics has been popular and successful for measuring text similarity (Socher et al., 2011a; Cheng and Kartsaklis, 2015; He et al., 2015; Kenter and de Rijke, 2015; Kusner et al., 2015; Ma et al., 2015; Tai et al., 2015; Wang et al., 2015). Word embeddings represent similar words in similar locations in vector space: "cat" is closer to "feline" than to "bird." It would be natural to expect such semantics-based approaches to be useful for clustering, particularly for corpora where authors have tried to express similar ideas in unique ways. And indeed, this paper will show that, depending on the choice of clustering method, semantics-based similarity measures such as summed word embeddings and deep neural networks can have an advantage over more traditional similarity metrics, such as n-gram counts, n-gram tf-idf vectors, and dependency tree kernels, when applied to creative texts.

However, unlike in most text similarity tasks, in clustering the choice of similarity metric interacts with both the choice of clustering method and the properties of the text. Graph-based clustering techniques can be quite effective in clustering short texts (Rangrej et al., 2011), yet this paper will show that they are sensitive to how tightly clustered the data is. Moreover, the tightness of clusters in a dataset is a property of both the underlying data and the similarity metric. We show that when the underlying data can be clustered tightly enough to use powerful graph-based clustering methods, using semantics-based similarity metrics actually creates a disadvantage compared to methods that rely on the surface form of the text, because semantic metrics reduce tightness.

The remainder of this paper is organized as follows. Section 4.1 summarizes related work. Section 4.2 describes four datasets of short texts. In Section 4.3, we describe the similarity metrics and clustering methods used in our experiments, as well as the evaluation measures. Section 4.4 shows that semantics-based similarity metrics have some advantage when clustering short texts from the most creative dataset, but ultimately do not perform the best when graph-based clustering is an option. In Section 4.5, we demonstrate the

powerful effect that tightness of clusters has on the best combination of similarity metric and clustering method for a given dataset. Finally, Section 4.6 draws conclusions.

4.1 Related Work

The most similar work to the present paper is Shrestha et al. (2012), which acknowledged that the similarity metric and the clustering method could both contribute to clustering results. It compared four similarity methods and also tested four clustering methods. Unlike the present work, it did not consider distributional semantics-based similarity measures or similarity measures that incorporated deep learning. In addition, it reported that the characteristics of the corpora "overshadow[ed] the effect of the similarity measures," making it difficult to conclude that there were any significant differences between the similarity measures.

Several papers address the choice of similarity metric for short text clustering without varying the clustering method. Yan et al. (2012) proposed an alternative term weighting scheme to use in place of tfidf when clustering using non-negative matrix factorization. King et al. (2013) used the cosine similarity between feature vectors that included context word and part-of-speech features and spelling features and applied Louvain clustering to the resulting graph. Xu et al. (2015) used a convolutional neural network to represent short texts and found that, when used with the k-means clustering algorithm, this deep semantic representation outperformed tf-idf, Laplacian eigenmaps, and average embeddings for clustering.

Other papers focused on choosing the best clustering method for short texts, but kept the similarity metric constant. Rangrej et al. (2011) compared k-means, singular value decomposition, and affinity propagation for tweets, finding affinity propagation the most effective, using tf-idf with cosine similarity or Jaccard for a similarity measure. Errecalde et al. (2010) describe an AntTree-based clustering method. They used the cosine similarity

of tf-idf vectors as well. Yin (2013) also use the cosine similarity of tf-idf vectors for a two-stage clustering algorithm for tweets.

One common strategy for short text clustering has been to take advantage of outside sources of knowledge (Banerjee et al., 2007; Wang et al., 2009a; Petersen and Poon, 2011; Rosa et al., 2011; Wang et al., 2014b). The present work relies only on the texts themselves, not external information.

4.2 Datasets

Collective discourse (Qazvinian and Radev, 2011; King et al., 2013) involves multiple writers generating texts in response to the same stimulus. In a corpus of texts relating to several stimuli, it may be desirable to cluster according to which stimulus each text relates to—for instance, grouping all of the news headlines about the same event together. Here, we consider texts triggered by several types of stimuli: photographs that need descriptive captions, cartoons that need humorous captions, and crossword answers that need original clues. Each need shapes the properties of the texts.

Pascal and Flickr Captions. The Pascal Captions dataset (hereinafter PAS) and the 8K ImageFlickr dataset (Rashtchian et al., 2010) are sets of captions solicited from Mechanical Turkers for photographs from Flickr and from the Pattern Analysis, Statistical Modeling, and Computational Learning (PASCAL) Visual Object Classes Challenge (Everingham et al., 2010).

PAS includes twenty categories of images (e.g., dogs, as in Example (1)) and 4998 captions. Each category has fifty images with approximately five captions for each image. We use the category as the gold standard cluster. The 8K ImageFlickr set includes 38,390 captions for 7663 photographs; we treat the image a caption is associated with as the gold standard cluster. To keep dataset sizes comparable, we use a subset of 5000 captions (998 clusters) from ImageFlickr (hereinafter FLK).

Figure 4.1: Image from PAS (Rashtchian et al., 2010)

(1) Captions for Figure 4.1:

"a man walking a small dog on a very wavy beach"

"A person in a large black coats walks a white dog on the beach through rough waves."

"Walking a dog on the edge of the ocean"

This task did not encourage creativity; instructions said to "describe the image in one complete but simple sentence." In addition, because photographs may contain overlapping elements—for instance, a photograph in the "bus" category of PAS might also show cars, while a photograph in the "cars" category could also contain a bus—these datasets should not be very tightly clustered.

***New Yorker* Cartoon Captions**. *The New Yorker* magazine has a weekly competition in which readers submit possible captions for a captionless cartoon (Example (1)) (Radev et al., 2015). We use the cartoon each caption is associated with as its gold standard cluster.

The complete dataset includes over 1.9 million captions for 366 cartoons. For this work,

Figure 4.2: Image from New Yorker cartoon captioning contest (Radev et al., 2015)

we use a total of 5000 captions from 20 randomly selected cartoons as the "TOON" dataset.

(2) Captions for Figure 4.2:

"Objection, Your Honor! Alleged killer whale."

"My client maintains that the penguin had a gun!"

"I demand a change of venue to a maritime court!"

Since caption writers seek to stand out from the crowd, we expect high creativity. This may encourage a more varied vocabulary than the FLK and PAS captions that merely describe the image. We also expect wide variation in the meanings of captions for the same cartoon, due to the different joke senses submitted for each. We therefore do not expect TOON to be tightly clustered.

Crossword Clues. A dataset of particularly creative texts is comprised of crossword

clues.[1] We use the clues as texts and the answer words as their gold standard cluster; all of the clues in Example (3) belong to the "toe" cluster.

(3) Part of the foot

Little piggy

tic-tac-___

The third O of OOO

The complete crossword clues dataset includes 1.7M different clues corresponding to 174,638 unique answers. The "CLUE" dataset includes 5000 clues corresponding to 20 unique answers selected by randomly choosing answers that have 250 or more unique clues, and then randomly choosing 250 of those clues for each answer.

Since words repeat, crossword authors must be creative to come up with clues that will not bore cruciverbalists. CLUE should thus contain many alternative phrasings for essentially the same idea. At the same time, there is likely to be relatively little overlap between clues for different answers, so CLUE should be tightly clustered.

4.3 Method

Here we describe the similarity metrics and clustering methods, as well as evaluation measures.

4.3.1 Similarity Metrics

We hypothesize that creative texts with wide vocabularies will benefit from similarity metrics based on semantic representation of the text, rather than its surface form. We there-

[1] Collected from http://crosswordgiant.com/

fore compare three metrics that rely on surface forms of words—n-gram count vectors, tf-idf vectors, and dependency tree segment counts—to three semantic ones—summed Word2Vec embeddings, LSTM autoencoders, and skip-thought vectors. In each case, we represent texts as vectors and find their cosine similarities; if cosine similarity can be negative, we add one and normalize by two to ensure similarity in the range [0, 1].

N-Gram Counts. First we consider n-gram count vectors. We use three variations: (1) unigrams, (2) unigrams and bigrams, and (3) unigrams, bigrams, and trigrams. The cosine similarity between u and v is given by Equation 4.1.

$$\text{Cosine Similarity}(u, v) = \frac{u \cdot v}{||u||\ ||v||} \tag{4.1}$$

N-Gram Tfidf. We also consider weighting n-grams by tf-idf, as calculated by sklearn (Pedregosa et al., 2011). Tf-idf is a common weighting scheme used in information retrieval that accounts for n-gram frequency, increasing the weight of words that are common in a particular document, but decreasing the weight of words that are common in the corpus as a whole. After weighting the vectors for sentences u and v, we again determine their similarity using Equation 4.1.

Dependency Counts. Grammatical information has been found to be useful in text, particularly short text, similarity. (Liu and Gildea, 2005; Zhang et al., 2005; Wang et al., 2009b; Heilman and Smith, 2010; Tian et al., 2010; Šarić et al., 2012; Tai et al., 2015). To leverage this information, previous work has used dependency kernels (Tian et al., 2010), which measure similarity by the fraction of identical dependency parse segments between two sentences. Here, we accomplish the same effect using a count vector for each sentence, with the dependency parse segments as the vocabulary. We define the set of segments for a dependency parse to consist of, for each word, the word, its parent, and the dependency relation that connects them as shown in Example (4).

(4) Part of shoe

a. Segment 1: (part, ROOT, nsubj)

b. Segment 2: (of, part, prep)

c. Segment 3: (shoe, of, pobj)

Word2Vec. For each word, we obtain, if possible, a vector learned via Word2Vec (Mikolov et al., 2013a) from the Google News corpus.[2] We represent a sentence as the normalized sum of its word vectors. We use the slightly modified cosine similarity measure in Equation 4.2 to ensure similarity in the range [0, 1].

$$\text{Modified Cosine Similarity}(u, v) = \frac{\frac{u \cdot v}{||u||\ ||v||} + 1}{2} \tag{4.2}$$

LSTM Autoencoder. We use Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) to build another semantics-based sentence representation. We train an LSTM autoencoder consisting of an encoder network and a decoder network. The encoder reads the input sentence and produces a single vector as the hidden state at the last time step. The decoder takes this hidden state vector as input and attempts to reconstruct the original sentence. The LSTM autoencoder is trained to minimize the reconstruction loss. After training, we extract the hidden state at the last time step of encoder as the vector representation for a sentence. Given the hidden state vectors for two sentences, again, the similarity score is computed using Equation 4.2.

Skip-thoughts (Kiros et al., 2015) trains encoder-decoder Recurrent Neural Networks (RNN) without supervision to predict the next and the previous sentences given the current sentence. The pretrained skip-thought model computes vectors as sentence representations. We then use Equation 4.2 to calculate the similarity between two sentences.

[2] https://code.google.com/archive/p/word2vec/

4.3.2 Clustering Methods

We explore five clustering methods: k-means, spectral, affinity propagation, Louvain, and MCL.

K-means is a popular and straightforward clustering algorithm (Berkhin, 2006) that takes a parameter k, the number of clusters, and uses an expectation-maximization approach to find k centroids in the data. In the expectation phase points are assigned to their nearest cluster centroid. In the maximization phase the centroids of are recomputed for each cluster of assigned points. K-means is not a graph-based clustering algorithm, but rather operates in a vector space.

Spectral clustering (Donath and Hoffman, 1973; Shi and Malik, 2000; Ng et al., 2001) is a graph-based clustering approach that finds the graph Laplacian of a similarity matrix, builds a matrix of the first k eigenvectors of the Laplacian, and then applies further clustering to this matrix. The method can be viewed as an approximation of a normalized min-cuts algorithm or of a random walks approach. We use the default implementation provided by sklearn, which applies a Gaussian kernel to determine the graph Laplacian and uses k-means for the subsequent clustering step.

Affinity propagation finds exemplars for each cluster and then assigns nodes to a cluster based on these exemplars (Frey and Dueck, 2007). This involves updating two matrices R and A, respectively representing the responsibility and availability of each node. A high value for $R_{(i,k)}$ indicates that node x_i would be a good exemplar for cluster k. A high value for $A_{(i,k)}$ indicates that node x_i is likely to belong to cluster k. We use the default implementation provided by sklearn.

Louvain initializes each node to be its own cluster, then greedily maximizes modularity (Section 4.5.1) by iteratively merging clusters that are highly interconnected (Blondel et al., 2008).

Markov Cluster Algorithm (MCL) simulates flow on a network via random walk (Van Dongen, 2000). The sequence of nodes is represented via a Markov chain. By

applying inflation to the transition matrix, the algorithm can maintain the cluster structure pronounced in the transition matrix of this random walk—a structure that would otherwise disappear over time.[3]

4.3.3 Evaluation Methods

Adjusted Rand Index We use the sklearn implementation of the Adjusted Rand Index (ARI)[4] (Hubert and Arabie, 1985):

$$\text{ARI} = \frac{RI - \text{Expected RI}}{\max RI - \text{Expected RI}} \tag{4.3}$$

where RI is the Rand Index,

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.4}$$

TP is the number of true positives, TN is true negatives, and FP and FN are false positives and false negatives, respectively. The Rand Index ranges from 0 to 1. ARI adjusts the Rand Index for chance, so that the score ranges from -1 to 1. Random labeling will achieve an ARI score close to 0; perfect labeling achieves an ARI of 1.

Purity is a score in the $[0, 1]$ range that indicates to what extent sentences in the same predicted cluster actually belong to the same cluster. Given $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$, the predicted clusters, $C = \{c_1, c_2, ..., c_J\}$, the true clusters, and N, the number of examples, purity is

$$\text{Purity}(\Omega, C) = \frac{1}{N} \sum_{k \in K} \max_{j \in J} |\omega_k \cap c_j| \tag{4.5}$$

Normalized Mutual Information (NMI). We use the sklearn implementation of NMI:

$$NMI(\Omega, C) = \frac{MI(\Omega, C)}{\sqrt{H(C) \cdot H(\Omega)}} \tag{4.6}$$

[3]We use the implementation from http://micans.org/mcl/ with inflation=2.0.
[4]Equivalent to Cohen's Kappa (Warrens, 2008).

The numerator is the mutual information (MI) of predicted cluster labels Ω and true cluster labels C. MI describes how much knowing what the predicted clusters are increases knowledge about what the actual classes are. Using marginal entropy ($H(x)$), NMI normalizes MI so that it ranges from 0 to 1. If C and Ω are identical—that is, if the clusters are perfect—NMI will be 1.

4.4 Vocabulary Width

4.4.1 Descriptive Statistics for Vocabulary Width

We predict that creative texts have a wider vocabulary than functional texts. We use two measures to reflect this wide vocabulary: the type/token ratio in the dataset (TTR), and that ratio normalized by the mean length of a text in the dataset.

TTR is an obvious estimate of the width of the vocabulary of a corpus. However, all other things being equal, a corpus of many very short texts triggered by the same stimulus would have more repeated words, proportional to the total number of tokens in the corpus, than would a corpus of a smaller number of longer texts. We might therefore normalize the ratio of types to tokens by dividing by the mean length of a text in the dataset, leading to the normalized type-to-token ratio (NTTR) and TTR values shown in Table 4.1.

	CLUE	TOON	PAS	FLK
TTR	0.1680	0.1064	0.0625	0.0561
NTTR	0.0377	0.0086	0.0058	0.0047

Table 4.1: Vocabulary properties of each dataset

FLK, PAS, and CLUE conform to expectations. The creative CLUE has TTR more than double that of the more functional PAS and FLK. The effect is more pronounced using NTTR. Surprisingly, TOON falls closer to the PAS and FLK end of the spectrum, suggesting that vocabulary width does not capture the creativity in the captioning competition; perhaps the creativity of cartoon captions is about expressing different ideas, rather than

finding unique ways to express the same idea. For the experiments based on vocabulary width, we therefore compare PAS and CLUE.

4.4.2 Experiments

We hypothesize that if a dataset uses a wide variety of words to express the same ideas, similarity metrics that rely on the surface form of the sentence will be at a disadvantage compared to similarity metrics based in distributional semantics. Thus, word2vec, LSTM autoencoders, and skip-thoughts ought to perform better than the n-gram-based methods and dependency count method when applied to CLUE, but should enjoy no advantage when applied to PAS.

We begin by comparing the performance of all similarity metrics on PAS and CLUE, using k-means for clustering. We then also examine their performance with MCL.

4.4.3 Results and Discussion

Table 4.2 compares the performance of all similarity metrics on PAS and CLUE using k-means and MCL. Using k-means on PAS, the unigram tfidf similarity metric gives the strongest performance for purity and NMI and came in a close second for ARI. LSTM slightly outperformed the other similarity metrics on ARI, but had middle-of-the-road results on the other evaluations. Overall, the semantics-based similarity metrics gave reasonable but not exceptional ARI and purity results, but were at the low end on NMI. For k-means on CLUE, the picture is quite different: the semantics-based similarity metrics markedly outperformed any other similarity metric on ARI. LSTM also provides the best purity score, followed by skip-thought. The semantics-based metrics do not stand out for NMI, though. Based on these results, we conclude that semantics-based measures provide a significant advantage over traditional similarity metrics when using k-means on the wide-vocabulary, creative CLUE.

k-Means

	PAS			CLUE		
Metric	ARI	Purity	NMI	ARI	Purity	NMI
Unigram	0.0286	0.141	0.110	0.0137	0.173	0.153
Bigram	0.0230	0.143	0.111	0.0124	0.165	0.142
Trigram	0.0289	0.139	0.108	0.0148	0.178	0.156
Uni. tfidf	0.0445	**0.189**	**0.169**	0.0180	0.202	0.188
Bi. tfidf	0.0287	0.158	0.135	0.0156	0.205	0.205
Tri. tfidf	0.0345	0.176	0.142	0.0134	0.195	**0.213**
Dependency	0.0122	0.131	0.104	0.0071	0.169	0.207
Word2Vec	0.0274	0.142	0.103	0.0527	0.189	0.165
LSTM	**0.0453**	0.170	0.142	**0.0837**	**0.240**	0.202
Skipthought	0.0311	0.140	0.106	0.0691	0.215	0.180

MCL

	PAS			CLUE		
Metric	ARI	Purity	NMI	ARI	Purity	NMI
Unigram	1.00E-05	0.058	0.051	0.0620	0.527	0.439
Bigram	2.50E-05	0.065	0.070	0.0835	0.585	0.465
Trigram	3.60E-05	0.069	0.081	0.1034	0.608	0.478
Uni. tfidf	2.20E-05	0.061	0.060	**0.1482**	0.643	0.506
Bi. tfidf	3.86E-04	0.104	0.135	0.1327	0.722	0.544
Tri. tfidf	6.49E-03	0.212	0.230	0.1280	**0.751**	**0.561**
Dependency	**2.07E-02**	**0.280**	**0.264**	0.0832	0.745	0.543
Word2Vec	0.000	0.050	0.000	0.0000	0.050	0.000
LSTM	0.000	0.050	0.000	0.0000	0.050	0.000
Skipthought	0.000	0.050	0.000	0.0000	0.050	0.0009

Table 4.2: A comparison of all similarity metrics on PAS and CLUE datasets, clustered using k-means and MCL. For all evaluations, higher scores are better.

When clustering with MCL, however, the semantics-based methods perform exceptionally poorly on both datasets. Interestingly, the n-gram-based similarity metrics performed very well when paired with MCL on CLUE—outperforming the best of the k-means scores—while the same metrics performed terribly with MCL on PAS.

We hypothesize that the semantics-based similarity metrics produce less tightly clustered data than the surface-form-based metrics do, and that this may make clustering difficult for some graph-based clustering methods. The next section describes how we test this hypothesis.

4.5 Tightness of Clusters

4.5.1 Descriptive Statistics for Tightness

Two pieces contribute to cluster tightness: the dataset itself and the choice of similarity metric. To illustrate, we represent each text with the vector for its similarity metric—for instance, the sum of its word2vec vectors or the unigram tfidf vector—and reduce it to two dimensions using linear discriminant analysis. We plot five randomly selected gold standard clusters. Plots for unigram tfidf and word2vec representations of PAS and CLUE are shown in Figures 4.3 and 4.4. These support the intuition that semantics-based similarity metrics are not as tightly clustered as n-gram-based metrics. Note also that the CLUE unigram tfidf clusters appear tighter than the PAS unigram tfidf clusters.

To quantify this, we compute modularity (Newman, 2004; Newman, 2006):[5]

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{4.7}$$

A_{ij} is the edge weight between nodes i and j. $\delta(c_i, c_j)$ indicates whether i and j belong

[5]Newman (2010) notes that modularity for even a perfectly mixed network generally cannot be 1 and describes a normalized modularity formula. We calculated both normalized and non-normalized modularity and found the pattern of results to be the same, so we report only modularity.

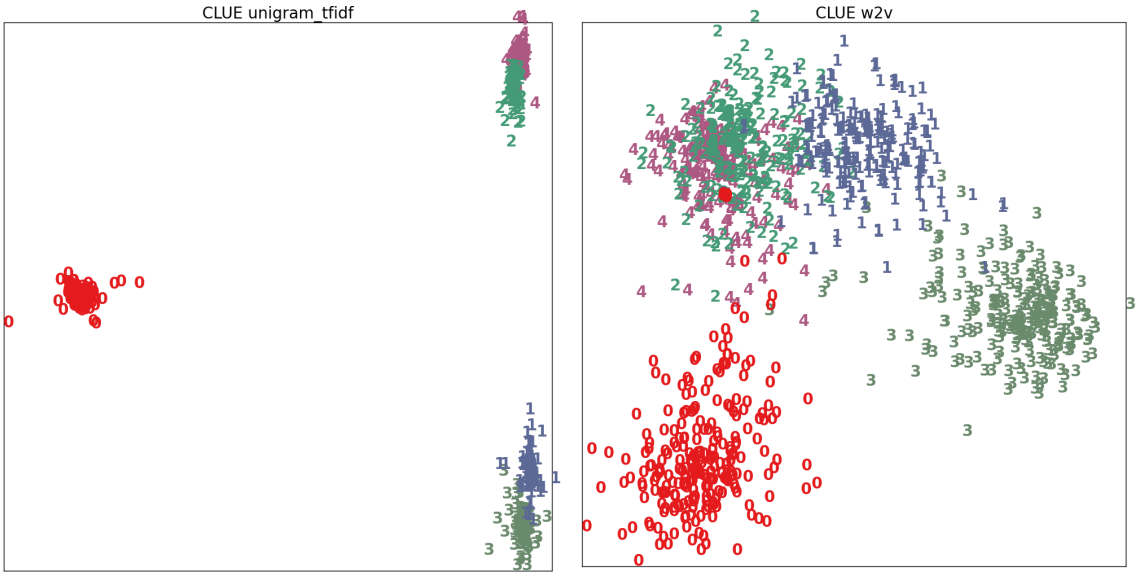


Figure 4.3: Plots of unigram tfidf (left) and word2vec (right) vectors representing five randomly selected clusters of CLUE: clues for words "ets," "stay," "yes," "easel," and "aha."

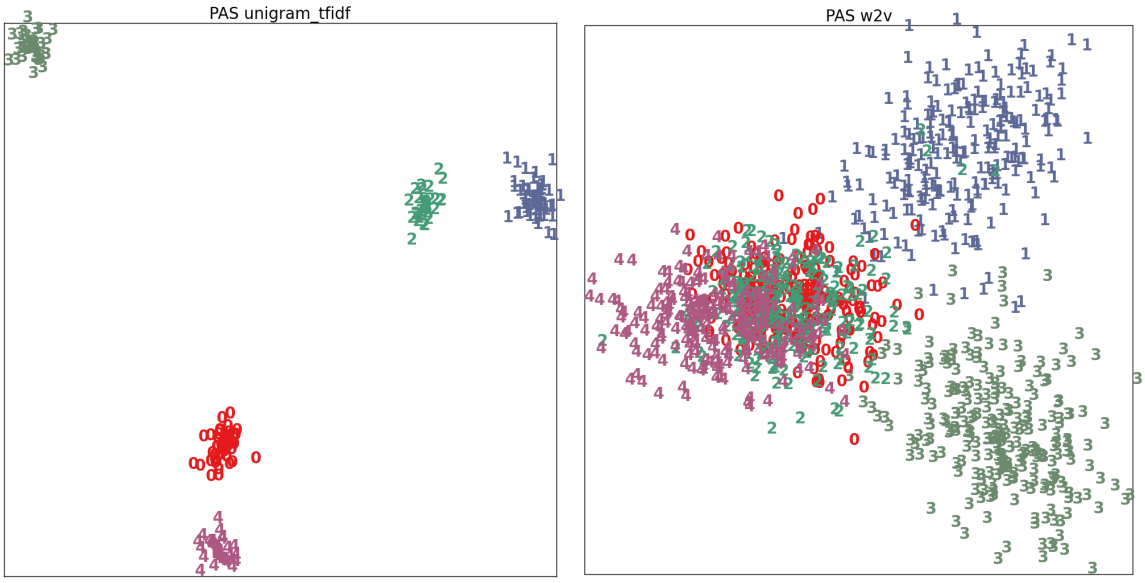


Figure 4.4: Plots of unigram tfidf (left) and word2vec (right) vectors representing five randomly selected clusters of PAS: images containing "bus," "boat," "car," "bird," and "motorbike."

to the same cluster. m is the number of edges. k_i is the degree of vertex i, so $\frac{k_i k_j}{2m}$ is the expected number of edges between i and j in a random graph. Thus, modularity is highest when nodes in a cluster are highly interconnected, but sparsely connected to nodes in dif-

ferent clusters. We use this statistic in an unconventional way, determining the modularity of the golden clusters.

Metric	PAS	Clues	TOON	FLK
Unigram	0.0254	0.1849	0.0214	0.0065
Trigram	0.0347	0.2447	0.0352	0.0135
Unigram tfidf	0.0587	0.3005	0.0519	0.0184
Trigram tfidf	0.0347	0.4339	**0.1311**	**0.0618**
Dependency	**0.0799**	**0.4729**	0.0451	0.0299
Word2Vec	0.0020	0.0036	0.0008	0.0004
Skipthought	0.0009	0.0028	0.0006	0.0003

Table 4.3: Modularity for all datasets

Table 4.3 shows the modularities for all four datasets using the unigram, trigram, unigram tfidf, trigram tfidf, dependency, word2vec, and skipthoughts similarity metrics. As suggested by Figures 4.3 and 4.4, the CLUE n-gram-based similarities have the highest modularity by far. The n-gram-based similarities for all datasets have much higher modularity than any of the semantics-based similarities; indeed, the semantics-based similarities rarely have modularity much higher than zero. Thus, we conclude both that CLUE is more tightly clustered than the other datasets and that n-gram-based measures yield tighter clusters than semantics-based measures.

To determine whether cluster tightness influences the best clustering method, we tested all clustering methods on all four datasets using unigram, trigram, unigram tfidf, trigram tfidf, word2vec, and skipthought similarity metrics.

4.5.2 Results and Discussion

As can be seen in Figure 4.5, the best ARI results by a large margin were those on the tightly clustered CLUE. Louvain, which provides the best ARI for CLUE, and MCL, which provides the second best, both performed most strongly when paired with the surface-form-based similarity metrics (n-gram counts, tfidf, and dependency count), which had high modularity relative to the semantics-based metrics. CLUE is also the only dataset

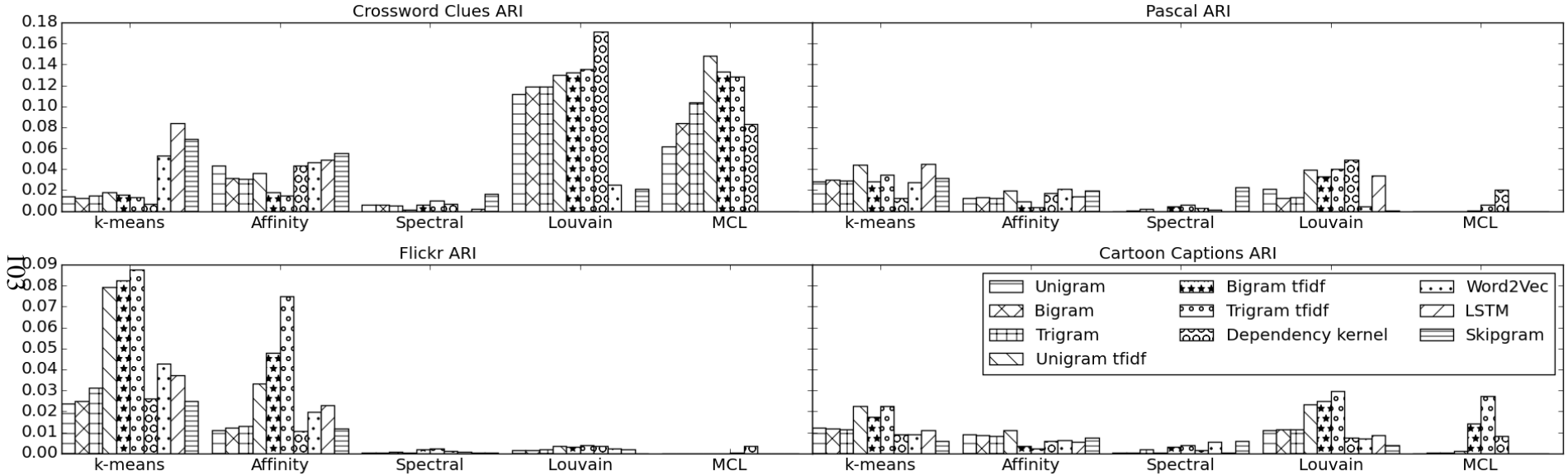


Figure 4.5: All similarity metrics and all clustering methods for the four datasets.

where the semantics-based similarity metrics performed exceptionally well with any of the clustering methods. As discussed earlier, this is likely due to the wider vocabulary in this dataset.

FLK, which had the lowest modularity, cannot be clustered by the spectral, Louvain, or MCL algorithms. K-means provides the strongest performance, followed by affinity propagation.

TOON has the worst ARI results. Its best-performing clustering methods are the graph-based Louvain and MCL methods. Both perform well only when paired with the most modular similarity metrics. Louvain seems less sensitive to modularity than MCL does. MCL's best performance by far for TOON is when it is paired with trigram tfidf, which also had the highest modularity; its performance when paired with the lower-modularity similarity metrics rapidly falls away. In contrast, Louvain fares reasonably well with the lower n-gram tfidfs, which also had lower modularity than trigram tfidf.

Louvain and MCL follow a similar pattern on PAS: both perform at their peak on the most modular similarity metric (dependency), but Louvain handles slightly less modular similarity metrics nearly as well as the most modular one, while MCL quickly falters.

K-means' performance is not correlated with modularity. This makes sense, as k-means is the only non-graph-based method. The fact that k-means nevertheless performs poorly on TOON suggests that this dataset may be particularly difficult to cluster. An interesting test would be to measure inter-annotator agreement on TOON.

4.6 Conclusions and Future Work

This work has shown that creativity can influence the best way to cluster text. When using k-means to cluster a dataset where authors tried to be creative, similarity metrics utilizing distributional semantics outperformed those that relied on surface forms. We also showed that semantics-based methods do not provide a notable advantage when applying k-means

to less creative datasets. Since traditional similarity metrics are often faster to calculate, use of slower semantics-based methods should be limited to creative datasets.

Unlike most work on clustering short texts, we examined how the similarity metric interacts with the clustering method. Even for a creative dataset, if the underlying data is tightly clustered, the use of semantics-based similarity measures can actually hurt performance. Traditional metrics applied to such tightly clustered data generate more modular output that enables the use of sophisticated, graph-based clustering methods such as MCL and Louvain. When either the underlying data or the similarity metrics applied to it produce loose clusters with low modularity, the sophisticated graph clustering algorithms fail, and we must fall back on simpler methods.

Future work can manipulate datasets' text properties to confirm that a specific property is the cause of observed differences in clustering. A pilot effort to use word embeddings to alter the variety of vocabulary in a dataset has so far not succeeded, but future experiments that altered vocabulary width or modularity of a dataset and found that the modified dataset behaved like natural datasets with the same properties could increase confidence in causality. Future work can also explore finer clusters within these datasets, such as clustering CLUE by word sense of the answers and TOON by joke sense.

These results are a first step towards determining the best way to cluster a new dataset based on properties of the text. Future work will explore further how the goals of short text authors translate into measurable properties of the texts they write, and how measuring those properties can help predict which similarity metrics and clustering methods will combine to provide the best performance.

Part II

From Text to Meaning Representation

CHAPTER 5

Improving Text-to-SQL Evaluation Methodology

5.1 Introduction

Effective natural language interfaces to databases (NLIDB) would give lay people access to vast amounts of data stored in relational databases. This chapter identifies key oversights in current evaluation methodology for this task. In the process, we (1) introduce a new, challenging dataset, (2) standardize and fix many errors in existing datasets, and (3) propose a simple yet effective baseline system.[1]

First, we consider query complexity, showing that human-written questions require more complex queries than automatically generated ones. To illustrate this challenge, we introduce *Advising*, a dataset of questions from university students about courses that lead to particularly complex queries.

Second, we identify an issue in the way examples are divided into training and test sets. The standard approach, shown at the top of Fig. 5.1, divides examples based on the text of each *question*. As a result, many of the queries in the test set are seen in training, albeit with different entity names and with the question phrased differently. This means metrics are mainly measuring robustness to the way a set of known SQL queries can be expressed in English—still a difficult problem, but not a complete test of ability to compose new queries in a familiar domain. We introduce a template-based slot-filling baseline that cannot

[1] Code and data is available at [for review see attachment].

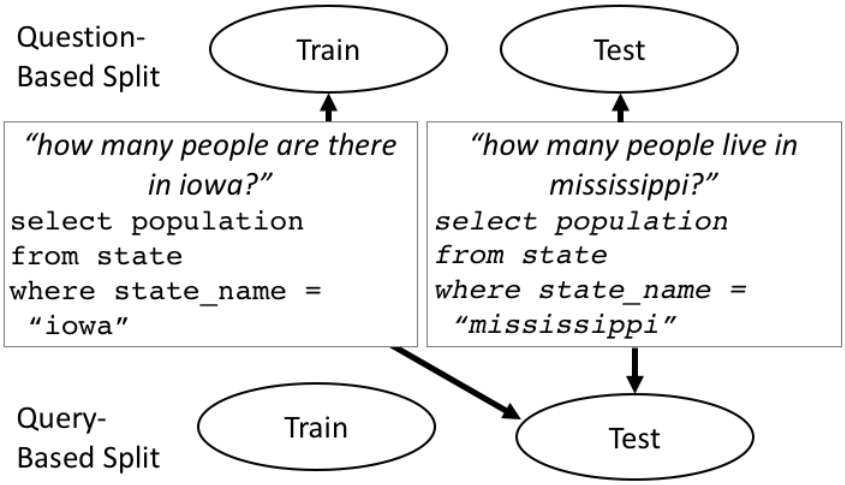


Figure 5.1: Traditional question-based splits allow queries to appear in both train and test. Our query-based split ensures each query is in only one.

generalize to new queries, and yet is competitive with prior work on multiple datasets. To measure robustness to new queries, we propose splitting based on the SQL *query*. We show that state-of-the-art systems with excellent performance on traditional question-based splits struggle on query-based splits. We also consider the common practice of variable anonymization, which removes a challenging form of ambiguity from the task. In the process, we apply extensive effort to standardize datasets and fix a range of errors.

Previous NLIDB work has led to impressive systems, but current evaluations provide an incomplete picture of their strengths and weaknesses. In this chapter, we provide new and improved data, a new baseline, and guidelines that complement existing metrics, supporting future work.

5.2 Related Work

The task of generating SQL representations from English questions has been studied in the NLP and DB communities since the 1970s (Androutsopoulos et al., 1995). Our observations about evaluation methodology apply broadly to the systems cited below.

Within the DB community, systems commonly use pattern matching, grammar-based

techniques, or intermediate representations of the query (Pazos Rangel et al., 2013). Recent work has explored incorporating user feedback to improve accuracy (Li and Jagadish, 2014). Unfortunately, none of these systems are publicly available and many rely on domain-specific resources.

In the NLP community, there has been extensive work on semantic parsing to logical representations that query a knowledge base (Zettlemoyer and Collins, 2005; Liang et al., 2011; Beltagy et al., 2014; Berant and Liang, 2014), while work on mapping to SQL has recently increased (Yih et al., 2015; Iyer et al., 2017; Zhong et al., 2017). One of the earliest statistical models for mapping text to SQL was the PRECISE system (Popescu et al., 2003; Popescu et al., 2004), which achieved high precision on queries that met constraints linking tokens and database values, attributes, and relations, but did not attempt to generate SQL for questions outside this class. Later work considered generating queries based on relations extracted by a syntactic parser (Giordani and Moschitti, 2012) and applying techniques from logical parsing research (Poon, 2013). However, none of these earlier systems are publicly available, and some required extensive engineering effort for each domain, such as the lexicon used by PRECISE.

More recent work has produced general purpose systems that are competitive with previous results and are also available, such as Iyer et al. (2017). We also adapt a logical form parser with a sequence to tree approach that makes very few assumptions about the output structure (Dong and Lapata, 2016).

One challenge for applying neural models to this task is annotating large enough datasets of question-query pairs. Recent work (Cai et al., 2017; Zhong et al., 2017) has automatically generated large datasets using templates to form random queries and corresponding natural-language-like questions, and then having humans rephrase the question into English. Another option is to use feedback-based learning, where the system alternates between training and making predictions, which a user rates as correct or not (Iyer et al., 2017). Other work seeks to avoid the data bottleneck by using end-to-end approaches (Yin

et al., 2016; Neelakantan et al., 2017), which we do not consider here. One key contribution of this chapter is standardization of a range of datasets, to help address the challenge of limited data resources.

5.3 Data

For our analysis, we modify a range of text-to-SQL datasets, standardizing them to have a consistent SQL style.

ATIS (Price, 1990; Dahl et al., 1994) User questions for a flight-booking task, manually annotated. We use the modified SQL from Iyer et al. (2017), which follows the data split from the logical form version (Zettlemoyer and Collins, 2007).

GeoQuery (Zelle and Mooney, 1996) User questions about US geography, manually annotated with Prolog. We use the SQL version (Popescu et al., 2003; Giordani and Moschitti, 2012; Iyer et al., 2017), which follows the logical form data split (Zettlemoyer and Collins, 2005).

Restaurants (Tang and Mooney, 2000; Popescu et al., 2003) User questions about restaurants, their food types, and locations.

Scholar (Iyer et al., 2017) User questions about academic publications, with automatically generated SQL that was checked by asking the user if the output was correct.

Academic (Li and Jagadish, 2014) Questions about the Microsoft Academic Search (MAS) database, derived by enumerating every logical query that could be expressed using a single page of the MAS website and writing sentences to match them. The domain is similar to that of Scholar, but their schemas differ.

Yelp and IMDB (Yaghmazadeh et al., 2017) Questions about the Yelp website and the Internet Movie Database, collected from colleagues of the authors who knew the type of information in each database, but not their schemas.

WikiSQL (Zhong et al., 2017) A large collection of automatically generated questions about individual tables from Wikipedia, paraphrased by crowd workers to be fluent English.

Advising (This Work) Our dataset of questions over a database of course information at an actual university, but with fictional student records. Some questions were collected from the EECS department Facebook page and others were written by CS students with knowledge of the database instructed to write questions they might ask in an academic advising appointment.

The authors manually labeled the initial set of questions with SQL. To ensure high quality, at least two annotators scored each question-query pair on a two-point scale for accuracy—did the query generate an accurate answer to the question?—and a three-point scale for helpfulness—did the answer provide the information the asker was probably seeking? Cases with low scores were fixed or removed from the dataset.

We collected paraphrases using Jiang et al. (2017)'s method, with manual inspection to ensure accuracy. For a given sentence, this produced paraphrases with the same named entities (e.g. course number EECS 123). To add variation, we annotated entities in the questions and queries with their types—such as course name, department, or instructor—and substituted randomly-selected values of each type into each paraphrase and its corresponding query. This combination of paraphrasing and entity replacement means an original question of "For next semester, who is teaching EECS 123?" can give rise to "Who teaches MATH 456 next semester?" as well as "Who's the professor for next semester's CHEM 789?"

5.3.1 SQL Canonicalization

SQL writing style varies. To enable consistent training and evaluation across datasets, we canonicalized the queries: (1) we alphabetically ordered fields in `SELECT`, tables in `FROM`, and constraints in `WHERE`; (2) we standardized table aliases in the form `<TABLE_NAME>alias<N>` for the `N`th use of the same table in one query; and (3) we standardized capitalization and spaces between symbols. We confirmed these changes do not alter the meaning of the queries via unit tests of the canonicalization code and manual inspection of the output. We also manually fixed some errors, such as ambiguous mixing of `AND` and `OR` (30 queries).

5.3.2 Variable Annotation

Existing SQL datasets do not explicitly identify which words in the question are used in the SQL query. Automatic methods to identify these variables, as used in prior work, do not account for ambiguities, such as words that could be either a city or an airport. To provide accurate anonymization, we annotated query variables using a combination of automatic and manual processing.

Our automatic process extracted terms from each side of comparison operations in SQL: one side contains quoted text or numbers, and the other provides a type for those literals. Often quoted text in the query is a direct copy from the question, while in some cases we constructed dictionaries to map common acronyms, like 'american airlines':`AA`, and times like '2pm':`1400`. The process flagged cases with ambiguous mappings, which we then manually processed. Often these were mistakes, which we corrected, such as missing constraints (e.g., 'papers in 2015' with no date limit in the query), extra constraints (e.g., limiting to a single airline despite no mention in the question), inaccurate constraints (e.g., 'more than 5' as `> 4`), and inconsistent use of 'this year' to mean different years in different queries.

	Sets Identified	Affected Queries
ATIS	141	380
GeoQuery	17	39
Scholar	60	152

Table 5.1: Manually identified duplicate queries

5.3.3 Query Deduplication

Three of the datasets had many duplicate queries (i.e., semantically equivalent questions with different SQL). To avoid this spurious ambiguity we manually grouped the data into sets of equivalent questions (Table 5.1). A second person manually inspected every set and ran the queries. Where multiple queries are valid, we kept them all, though only used the first for the rest of this work.

5.4 Dataset Characteristics Show Evaluating on Multiple Datasets Is Necessary

For evaluation to be informative it must use data that is representative of real-world queries. If datasets have biases, robust comparisons of models will require evaluation on multiple datasets. For example, some datasets, such as ATIS and Advising, were collected from users and are task-oriented, while others, such as WikiSQL, were produced by automatically generating queries and engaging people to express the query in language. If these two types of datasets differ systematically, evaluation on one may not reflect performance on the other. In this section, we provide descriptive statistics aimed at understanding how several datasets differ, especially with respect to query redundancy and complexity.

```
SELECT <table-alias>.<field>
FROM <table> AS <table-alias>
WHERE <table-alias>.<field> = <literal>
```

```
SELECT RIVERalias0.RIVER_NAME
FROM RIVER AS RIVERalias0
WHERE RIVERalias0.TRAVERSE = "florida";
```

```
SELECT CITYalias0.CITY_NAME
FROM CITY AS CITYalias0
WHERE CITYalias0.STATE_NAME = "alabama";
```

Figure 5.2: An SQL pattern and example queries.

5.4.1 Measures

We consider a range of measures that capture different aspects of data complexity and diversity:

Question / Unique Query Counts We measure dataset size and how many distinct queries there are when variables are anonymized. We also present the mean number of questions per unique query; a larger mean indicates greater redundancy.

SQL Patterns Complexity can be described as the answer to the question, "How many query-form patterns would be required to generate this dataset?" Fig. 5.2 shows an example of a pattern, which essentially abstracts away from the specific table and field names. Some datasets were generated from patterns similar to these, including WikiSQL and Cai et al. (2017). This enables the generation of large numbers of queries, but limits the variation between them to only that encompassed by their patterns. We count the number of patterns needed to cover the full dataset, where larger numbers indicate greater diversity. We also report mean queries per pattern; here, larger numbers indicate greater redundancy, showing that many queries fit the same mold.

Counting Tables We consider the total number of tables and the number of unique tables mentioned in a query. These numbers differ in the event of self-joins. In both cases, higher values imply greater complexity.

	Question count	Unique query count	Questions per unique query	Queries per pattern μ	Queries per pattern Max	Pattern count
Advising	3898	208	18.7	15.6	41	188
ATIS	5280	947	5.6	7.0	870	751
GeoQuery	877	247	3.6	9.3	340	71
Restaurants	378	23	16.4	22.2	81	17
Scholar	817	193	4.2	5.6	71	146
Academic	196	185	1.1	2.1	12	92
IMDB	131	89	1.5	2.5	21	52
Yelp	128	110	1.2	1.4	11	89
WikiSQL	80,654	77,840	1.0	165.3	42,816	488

Table 5.2: Descriptive statistics for text-to-SQL dataset diversity. Datasets in the first group are human-generated from the NLP community, in the second are human-generated from the DB community, and in the third are automatically-generated.

Nesting A query with nested subqueries may be more complex than one without nesting. We count SELECT statements within each query to determine the number of sub-queries. We also report the depth of query nesting. In both cases, higher values imply greater complexity.

BLEU We use BLEU scores as an approximation of how similar questions in the dataset are, and how similar queries are. Higher scores would suggest less variation, and thus lower complexity. However, we did not observe any informative patterns and so omit these results.

5.4.2 Analysis

The statistics in Tables 5.3 and 5.2 show several patterns.

First, Table 5.2 shows that dataset size is not the best indicator of dataset diversity. Although WikiSQL contains fifteen times as many question-query pairs as ATIS, ATIS contains significantly more patterns than WikiSQL; moreover, WikiSQL's queries are dominated by one pattern that is more than half of the dataset (`SELECT col AS result FROM`

	Tables per query		Unique tables per query		SELECTs per query		Nesting Depth	
	μ	Max	μ	Max	μ	Max	μ	Max
Advising	3.2	9	3	8	1.2	6	1.2	4
ATIS	6.4	32	3.8	12	1.79	8	1.39	8
GeoQuery	1.4	5	1.1	4	2.17	8	2.03	7
Restaurants	2.6	5	2.3	4	1.17	2	1.17	2
Scholar	3.3	6	3.2	6	1.02	2	1.02	2
Academic	3.2	10	3	6	1.04	3	1.04	2
IMDB	1.9	5	1.9	5	1.01	2	1.01	2
Yelp	2.2	4	2	4	1	1	1	1
WikiSQL	1	1	1	1	1	1	1	1

Table 5.3: Descriptive statistics for text-to-SQL dataset complexity. Datasets in the first group are human-generated from the NLP community, in the second are human-generated from the DB community, and in the third are automatically-generated.

`table WHERE col = value`). The small, hand-curated datasets developed by the database community—Academic, IMDB, and Yelp—have noticeably less redundancy as measured by questions per unique query and queries per pattern than the datasets the NLP community typically evaluates on.

Second, Table 5.3 shows that human-generated datasets exhibit greater complexity than automatically generated data. All of the human-generated datasets except Yelp demonstrate at least some nesting. The average query from any of the human-generated datasets joins more than one table.

In particular, task-oriented datasets require joins and nesting. ATIS and Advising, which were developed with air-travel and student-advising tasks in mind, respectively, both score in the top three for multiple complexity scores.

To accurately predict performance on human-generated or task-oriented questions, it is thus necessary to evaluate on datasets that test the ability to handle nesting and joins. Training and testing NLP systems, particularly deep learning-based methods, benefits from large datasets. However, at present, the largest dataset available does not provide the desired complexity.

Takeaway: Evaluate on multiple datasets, some with nesting and joins, to provide a thorough picture of a system's strengths and weaknesses.

5.5 Current Data Splits Only Partially Probe Generalizability

It is standard best practice in machine learning to divide data into disjoint training, development, and test sets. Otherwise, evaluation on the test set will not accurately measure how well a model generalizes to new examples. The standard splits of GeoQuery, ATIS, and Scholar treat each pair of a natural language question and its SQL as a single item. Thus, as long as each question-query pair appears in only one set, the test set is not tainted with training data. We call this a question-based split.

However, many English questions may correspond to the same SQL query. If at least one copy of every SQL query appears in training, then the task evaluated is classification, not true semantic parsing, of the English questions. We can increase the number of distinct SQL queries by varying what entities our questions ask about; the queries for "*what states border Texas*" and "*what states border Massachusetts*" are not identical. Adding this variation changes the task from pure classification to classification plus slot-filling. Does this provide a true evaluation of the trained model's performance on unseen inputs?

It depends on what we wish to evaluate. If we want a system that answers questions within a particular domain, and we have a dataset that we are confident covers everything a user might want to know about that domain, then evaluating on the traditional question-based split tells us whether the system is robust to variation in how a request is expressed. But compositionality is an essential part of language, and a system that has trained on "*What courses does Professor Smith teach?*" and "*What courses meet on Fridays?*" should be prepared for "*What courses that Professor Smith teaches meet on Fridays?*" Evaluation

on the question split does not tell us about a model's generalizable knowledge of SQL, or even its generalizable knowledge within the present domain.

To evaluate the latter, we propose a complementary new division, where no SQL query is allowed to appear in more than one set; we call this the *query split*. To generate a query split, we substitute variables for entities in each query in the dataset, as described in § 5.3.2. Queries that are identical when thus anonymized are treated as a single query and randomly assigned—with all their accompanying questions—to train, dev, or test. We include the original question split and the new query split labeling for the new Advising dataset, as well as ATIS, GeoQuery, and Scholar. For the much smaller Academic, IMDB, Restaurant, and Yelp datasets, we include question- and query- based buckets for cross validation.

5.5.1 Systems

Recently, a great deal of work has used variations on the seq2seq model. We compare performance of a basic seq2seq model (Sutskever et al., 2014), and seq2seq with attention over the input (Bahdanau et al., 2015), implemented with TensorFlow seq2seq (Britz et al., 2017). We also extend that model to include an attention-based copying option, similar to Jia and Liang (2016). Our output vocabulary for the decoder includes a special token, `COPY`. If `COPY` has the highest probability at step t, we replace it with the input token with the max of the normalized attention scores. Our loss function is the sum of two terms. First, the categorical cross entropy for the model's probability distribution over the output vocabulary tokens. Second, the loss for word copying. When the correct output token is `COPY`, the second loss term is the categorical cross entropy of the distribution of attention scores at time t. Otherwise it is zero.

For comparison, we include systems from two recent papers. Dong and Lapata (2016) used an attention-based seq2tree model for semantic parsing of logical forms; we apply their code here to SQL datasets. Iyer et al. (2017) use a seq2seq model with automatic

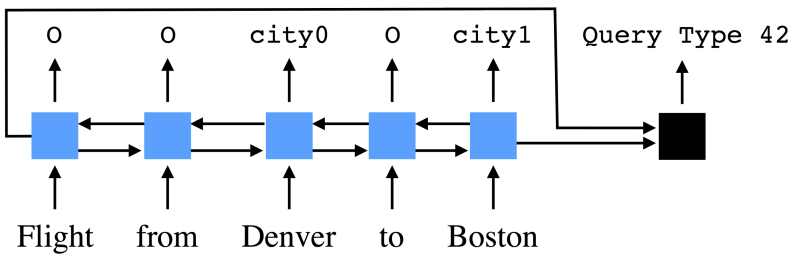


Figure 5.3: Baseline: blue boxes are LSTM cells and the black box is a feed-forward network. Outputs are the query template to use (right) and which tokens to fill it with (left).

dataset expansion through paraphrasing and SQL templates.[2]

We could not find publicly available code for the non-neural text-to-SQL systems discussed in Section 5.2. Also, most of those approaches require development of specialized grammars or templates for each new dataset they are applied to, so we do not compare such systems.

5.5.1.1 Template Baseline

In addition to the seq2seq models, we develop a new baseline system for text-to-SQL parsing which exploits repetitiveness in data. First, we automatically generate SQL templates from the training set. The system then makes two predictions: (1) which template to use, and (2) which words in the sentence should fill slots in the template. This system is not able to generalize beyond the queries in the training set, so it will fail completely on the new query-split data setting.

Fig. 5.3 presents the overall architecture. A bidirectional LSTM provides a prediction for each word, either `O` if the word is not used in the final query, or a symbol such as `city1` to indicate that it fills a slot. The hidden states of the LSTM at each end of the sentence are passed through a small feed-forward network to determine the SQL template to use. This architecture is simple and enables a joint choice of the tags and the template, though we do

[2] We enable Iyer et al. (2017)'s paraphrasing data augmentation, but not their template-based augmentation because templates do not exist for most of the datasets (though they also found it did not significantly improve performance). Note, on ATIS and Geo their evaluation assumed no ambiguity in entity identification, which is equivalent to our Oracle Entities condition (§5.5.2).

	Adv.		ATIS		Geo.		Rest.		Scholar		Ac.		IMDB		Yelp	
Model	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q	?	Q
No Variable Anonymization																
Baseline	**80**	0	46	0	56	0	95	0	52	0	0	0	0	0	1	0
seq2seq	4	0	8	0	32	3	47	0	19	0	6	7	1	0	0	0
+ Attention	26	0	46	18	54	22	100	2	33	0	71	64	7	3	2	2
+ Copying	74	0	**51**	**32**	**69**	31	100	4	**59**	5	**81**	**74**	**26**	**9**	**12**	4
D&L seq2tree	42	0	46	23	62	28	100	**11**	44	6	63	54	6	2	1	2
Iyer et al.	37	0	45	17	65	**37**	100	8	44	3	76	70	10	4	6	**6**
With Oracle Entities																
Baseline	90	0	56	0	58	0	95	0	66	0	0	0	7	0	8	0
seq2seq	18	0	14	0	56	4	71	6	23	0	10	9	6	0	12	9
+ Attention	90	0	57	23	71	29	100	32	71	4	77	74	44	17	33	28
D&L seq2tree	89	0	56	34	66	29	100	21	68	6	65	61	36	10	26	23
Iyer et al.	83	0	58	32	70	41	100	33	71	1	77	75	52	24	44	32
Baseline-Oracle	99	0	69	0	78	0	100	0	84	0	11	0	47	0	25	0

Table 5.4: Accuracy of neural text-to-SQL systems on English question splits ('?' columns) and SQL query splits ('Q' columns). The vertical line separates datasets from the NLP (left) and DB (right) communities. Results for Iyer et al. (2017) are slightly lower here than in the original paper because we evaluate on SQL output, not the database response. [Abbreviations: Adv. = Advising, Geo. = GeoQuery, Rest. = Restaurants, Ac. = Academic]

not explicitly enforce agreement.

To train the model, we automatically construct a set of templates and slots. Slots are determined based on the variables in the dataset, with each SQL variable that is explicitly given in the question becoming a slot. We can construct these templates because our new version of the data explicitly defines all variables, their values, and where they appear in both question and query.

For completeness, we also report on an oracle version of the template-based system (performance if it always chose the correct template from the train set and filled all slots correctly).

5.5.2 Oracle Entity Condition

Some systems, such as Dong and Lapata's model, are explicitly designed to work on anonymized data (i.e., data where entity names are replaced with a variable indicating

their type). Others, such as attention-based copying models, treat identification of entities as an inextricable component of the text-to-SQL task. We therefore describe results on both the actual datasets with entities in place and a version anonymized using the variables described in § 5.3.2. We refer to the latter as the oracle entity condition.

5.5.3 Results and Analysis

		Advising		ATIS		GeoQuery		Scholar	
		?	Q	?	Q	?	Q	?	Q
Correct	Count	369	5	227	111	191	56	129	17
	μ Length	83.8	165.8	55.1	69.2	19.6	21.5	38.0	30.2
Entity problem	Count	10	0	1	6	5	0	5	0
	μ Length	111.8	N/A	28.0	71.3	17.2	N/A	42.6	N/A
Different template	Count	43	675	94	68	53	84	40	94
	μ Length	69.8	68.4	85.8	72.1	25.6	18.0	43.9	39.8
No template match	Count	79	25	122	162	30	42	44	204
	μ Length	88.8	90.5	113.8	92.2	29.7	25.0	42.1	41.6

Table 5.5: Types of errors by the attention-based copying model for question and query splits, with (Count)s of queries in each category, and the (μ Length) of gold queries in the category.

We hypothesized that even a system unable to generalize can achieve good performance on question-based splits of datasets, and the results in Table 5.4 substantiate that for the NLP community's datasets. The template-based, slot-filling baseline was competitive with state-of-the-art systems for question split on the four datasets from the NLP community. The template-based oracle performance indicates that anywhere from 70-100% accuracy on question-based split could be obtained by selecting a template from train and filling in the slots for these datasets.

For the three datasets developed by the databases community, the effect of question-query split is far less pronounced. The small sizes of these datasets cannot account for the difference, since even the oracle baseline did not have much success on these question splits, and since the baseline was able to handle the small Restaurants dataset. Looking

back at Section 5.4, however, we see that these are the datasets with the least redundancy in Table 5.2. Because of their nearly 1:1 question:unique-query ratios, the question splits and query splits of these datasets were quite similar.

Reducing redundancy does not improve performance on query split, though; at most, it reduces the difference between performance on the two splits. IMDB and Yelp both show weak results on query split despite their low redundancy. Experiments on a non-redundant version of query split for Advising, ATIS, GeoQuery, and Restaurant that contained only one question for each query confirmed this: in each case, accuracy remained the same or declined relative to regular query split.

Having ruled out redundancy as a cause for the exceptional performance on Academic's query split, we suspect the simplicity of its questions and the compositionality of its queries may be responsible. Every question in the dataset begins "return me," followed by a phrase indicating the desired field, optionally followed by one or more constraints; for instance, *"return me the papers by 'author_name0'"* and *"return me the papers by 'author_name0' on journal_name0."*

None of this, of course, is to suggest that question-based split is an easy problem, even on the NLP community's datasets. Except for the Advising and Restaurants datasets, even the oracle version of the template-based system is far from perfect. Access to oracle entities helps performance of non-copying systems substantially, as we would expect. Entity matching is thus a non-trivial component of the task.

But the query-based split is certainly more difficult than the question-based split. Across datasets and systems, performance suffered on query split. Access to oracle entities did not remove this effect.

Many of the seq2seq models do show some ability to generalize, though. Unlike the template-based baseline, many were able to eek out some performance on query split.

On question split, ATIS is the most difficult of the NLP datasets, yet on query split, it is among the easiest. To understand this apparent contradiction, we must consider what kinds

of mistakes systems make and the contexts in which they appear. We therefore analyze the output of the attention-based-copying model in greater detail.

We categorize each output as shown in column one of Table 5.5. The "Correct" category is self-explanatory. "Entity problem only" means that the query would have been correct but for a mistake in one or more entity names. "Different template" means that the system output was the same as another query from the dataset but for the entity names; however, it did not match the correct query for this question. "No template match" contains both the most mundane and the most interesting errors. Here, the system output a query that is not copied from training data. Sometimes, this is a simple error, such as inserting an extra comma in the `WHERE` clause. Other times, it is recombining segments of queries it has seen into new queries. This is necessary but not sufficient model behavior in order to do well on the query split. In at least one case, this category includes a semantically equivalent query marked as incorrect by the exact-match accuracy metric. Table 5.5 shows the number of examples from the test set that fell into each category, as well as the mean length of gold queries ("length") for each category.

Short queries are easier than long ones in the question-based condition. In most cases, length in "correct" is shorter than length in either "different template" or "no template match" categories.

In addition, for short queries, the model seems to prefer to copy a query it has seen before; for longer ones, it generates a new query. In every case but one, mean length in "different template" is less than in "No template match."

Interestingly, in ATIS and GeoQuery, where the model performs tolerably well on query split, the length for correct queries in query split is higher than the length for correct queries from the question split. Since, as noted above, recombination of template pieces (as we see in "no template match") is a necessary step for success on query split, it may be that longer queries have a higher probability of recombination, and therefore a better chance of being correct in query split. The data from Scholar does not support this position; however,

note that only 17 queries were correct in Scholar query split, suggesting caution in making generalizations from this set.

These results also seem to indicate that our copying mechanism effectively deals with entity identification. Across all datasets, we see only a small number of entity-problem-only examples. However, comparing the rows from Table 5.4 for seq2seq+Copy at the top and seq2seq+Attention in the oracle condition, it is clear that having oracle entities provides a useful signal, with consistent gains in performance.

Takeaways: Evaluate on both question-based and query-based dataset splits. Additionally, variable anonymization noticeably decreases the difficulty of the task; thus, thorough evaluations should include results on datasets without anonymization.

5.5.4 Logic Variants

To see if our observations on query and question split performance apply beyond SQL, we also consider the logical form annotations for ATIS and GeoQuery (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007). We retrain Jia and Liang (2016)'s baseline and full system. Interestingly, we find limited impact on performance, measured with either logical forms or denotations. To understand why, we inspect the logical form datasets. In both ATIS and GeoQuery, the logical form version has a larger set of queries after variable identification. This seems to be because logic abstracts away from surface form less than SQL does. For example, these questions get the same SQL, but different logical forms:

"what state has the largest capital"

```
(A, (state(A), loc(B, A), largest(B, capital(B))))
```

"which state 's capital city is the largest"

```
(A, largest(B, (state(A), capital(A, B), city(B))))
```

By being closer to a syntactic representation, the queries end up being more compositional, and forcing the model to learn more compositionality than the SQL systems do.

5.6 Conclusion

In this work, we identify two issues in current datasets for mapping questions to SQL queries. First, by analyzing question and query complexity we find that human-written datasets require properties that have not yet been included in large-scale automatically generated query sets. Second, we show that the generalizability of systems is overstated by the traditional data splits. In the process we also identify and fix hundreds of mistakes across multiple datasets and homogenize the SQL query structures to enable effective multi-domain experiments.

Our analysis has clear implications for future work. Evaluating on multiple datasets is necessary to ensure coverage of the types of questions humans generate. Developers of future large-scale datasets should incorporate joins and nesting to create more human-like data. And new systems should be evaluated on both question- and query- based splits, guiding the development of truly general systems for mapping natural language to structured database queries.

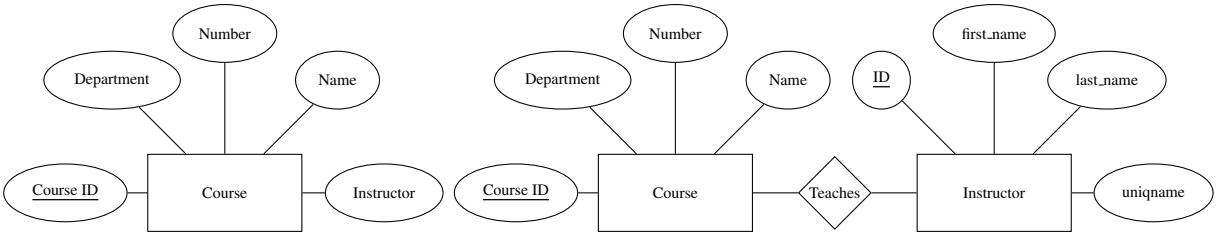
CHAPTER 6

Schema-Aware Text-to-SQL

6.1 Introduction

The ability to automatically transform English text to SQL queries could make a great deal of information more readily available. It would enable lay-people to get answers from relational databases without needing to learn SQL. It would also enable task-oriented dialog systems to easily incorporate relational databases. Recent research into this area (Iyer et al., 2017) has focused on sequence-to-sequence (seq2seq) models, which have shown promise in other semantic parsing tasks (Dong and Lapata, 2016). In the present work, we investigate modifications to the seq2seq model that incorporate knowledge of the relational database schema.

SQL queries are run over databases with formal schemas, and the schema of a particular database determines which queries are valid for that database. For example, consider



(a) Single Table Schema (b) Multi-Table Schema

Figure 6.1: Two possible schemas for a database that could answer, "Who teaches Discrete Mathematics?"

using a database of courses and instructors to answer the question, "Who teaches Discrete Mathematics?" If we are querying a database with the schema in Figure 6.1a, an appropriate query could be

```
SELECT instructor
FROM COURSE
WHERE name LIKE "Discrete Mathematics"
```

However, in a schema like that of Figure 6.1b, we would need the query to be

```
SELECT i.first_name, i.last_name
FROM INSTRUCTOR i, COURSE c
WHERE i.instructor_id = c.instructor_id
AND c.name LIKE "Discrete Mathematics"
```

In light of the importance of schema to queries, we hypothesize that a model that incorporates an explicit representation of a database's schema will have an advantage over one that must learn a hidden representation. We therefore introduce a semantic parsing model that incorporates neural attention to the database schema. Moreover, a model that can learn to copy field and table names from schemas into queries has the potential to learn about SQL from domains where data is plentiful and, provided it is given a schema, generalize to a new domain.

In the present work, we describe new neural architectures that incorporate explicit representations of schemas. We introduce two schema representations. We report performance of these models on both question-based and query-based splits (see Chapter 5) of the Advising, ATIS, GeoQuery, and Scholar datasets. In addition, unlike previous work, we report results of cross-domain models; that is, we train a model using the training data from all four of our datasets, and report its performance on the test data for each dataset. We conclude with proposals for future work building on our findings.

6.2 Related Work

Transforming text to SQL has been a goal of both the databases community and the NLP community for decades (Androutsopoulos et al., 1995; Pazos Rangel et al., 2013).

The PRECISE system (Popescu et al., 2003; Popescu et al., 2004) achieved, as the name suggests, high precision, but only on a class of "semantically tractable" questions. Such questions needed to obey detailed rules regarding correspondences between tokens representing database values, attributes, and relations. PRECISE did not attempt to generate for questions outside this class, such as"List flights from Oakland to Salt Lake City leaving after midnight Thursday." In addition, the system required a large lexicon to be built for every new database.

A number of systems use domain-specific resources. Giordani and Moschitti (2012) used lexical dependencies from the question and metadata from the database to build plausible `SELECT`, `FROM`, and `WHERE` clauses. They then combined the clauses using handwritten rules and heuristics, giving ranked candidate queries, and used a tree-kernel SVM-reranker to choose the best of these. Poon (2013) described an unsupervised semantic parser that first translates from NL to a "semantic tree" MR, then deterministically converts that MR to SQL. Rather than rely on an abundance of training examples, this system, too, required domain-specific human effort, in the form of creating domain-dependent states. Li and Jagadish (2014) described a system that interacts with the user to ensure proper interpretation of the question. It used a parse tree node mapper to map nodes from the dependency parse of the English sentence to SQL components, generating a "query tree." It then showed the query tree to the user, allowing the user to verify that the interpretation is correct. Mapping the parse tree to SQL components required enumerated sets of phrases for types of nodes; to map names and values from the input question to the database, they used a similarity function incorporating WordNet and spelling similarity. Saha et al. (2016) used domain-specific ontologies to describe the semantic entities within the domain and their relationships, as well as an ontology-to-database mapping that describes how ontology elements

map to database elements. Thus, the system first generates a query in an intermediate query language over the ontology, and then generates SQL from that. For domains in which they have built ontologies, they achieve perfect or near-perfect precision and recall of over 85%.

Neural systems generally try to avoid requiring human-written domain-specific resources, relying instead on access to a large number of training examples.

The most similar related work is Zhong et al. (2017)'s seq2SQL system, which uses an augmented pointer network, enabling it to perform reasonably well on simple queries across domains. The system is designed to work only on single table queries with no nesting. Input to its encoder is a concatenation of all column names in that single table, the question, and a limited vocabulary of SQL keywords such as `SELECT` and `COUNT`. The decoder is comprised of three parts. A classifier selects an aggregation operator (or null). A second classifier chooses one of the column names from the input sequence to go into the `SELECT` clause. And a pointer network generates the `WHERE` clause by copying from the input sequence. The two classifiers use cross entropy as a loss function, but the `WHERE` clause network uses a policy gradient, since the order of conditions can vary without altering the query's meaning.

Our work differs from seq2SQL in several important ways in order to permit queries that join multiple tables and queries with nesting. As noted in Chapter 5, joins and nesting are essential parts of human-generated SQL datasets. This has several implications for our network. First, the network needs to have access to table names, not just column names, from the schema. Second, the three-part structure of the seq2SQL decoder cannot apply, since it does not include a component for the `FROM` clause. Simply adding a fourth component to output the `FROM` clause would not fix the problem, since the network still would not have a way to handle nested queries. We therefore output the query as a sequence of tokens. We also handle the permitted variation in `WHERE` clause ordering differently. Rather than using a reinforcement-learning approach, we canonicalize queries. Specifically, we order the components of the `WHERE` clause alphabetically for all training inputs.

Iyer et al. (2017) also used seq2seq models for the text-to-SQL task. However, their approach did not take the database schema into account. Nor did they use copying from either input or schema.

Cai et al. (2017) use logic external to the seq2seq network to provide additional information to the network. In the encoder phase, they append a few new bits into the LSTM indicating whether the word is likely to be a table name or a string value. In the decoder network, they add a hidden state that they call the grammatical state that guarantees grammatical correctness of output. They report results on two datasets that they developed, which are not publicly available. The queries include zero or one table joins and one constraint in the `WHERE` clause.

Several systems have bypassed SQL as a representation, focusing instead on an end-to-end approach. This includes Neural Enquirer (NE) (Yin et al., 2016), neural network designed to execute natural language queries over a knowledge base. It represents the knowledge base as a table and learns embeddings, where each embedding represents a value in the table. Although NE uses word embeddings for field names in the table, it differs from the present work in several ways. First, NE does not generate SQL queries. Rather, its architecture incorporates a number of "executor" layers, each of which represents an operation, such as *select* or *max*, which returns intermediate results. Each executor operates on each row of the table. Second, NE requires an embedding for every value in the KB. Thus, the synthetic task on which it was evaluated used a table size of 10 by 10; the present work does not limit table size. In addition, NE is designed to work with one table. When required to join two tables, its accuracy declined by more than 18% (Yin et al., 2015).

6.3 Methods

Our systems are modified versions of seq2seq models that incorporate attention to schema.

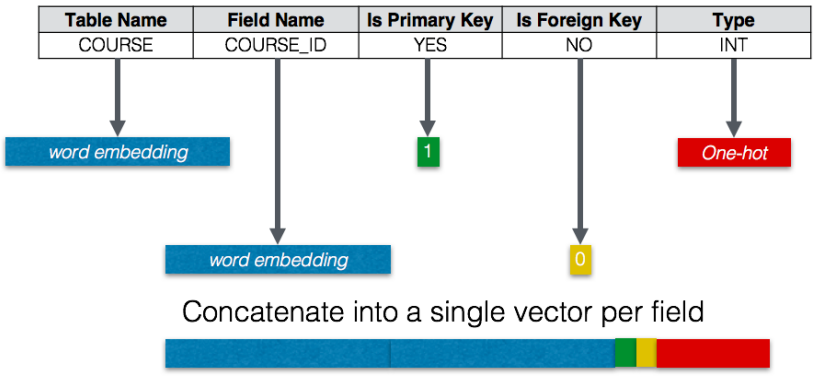


Figure 6.2: Schema embedding for COURSE.COURSE_ID.

6.3.1 Schema Representations

In order to pay attention to schema, we require a matrix representation of a database schema. We experiment with two such representations: schema embeddings and schema maps.

For schema embeddings, we generate fixed embeddings to represent each table and field in the schema. For each field, we first collect the table name, field name, and variable type, as well as whether it is a primary key and/or a foreign key. We use 50-dimensional GloVe word embeddings pre-trained on Wikipedia and Gigaword[1] to represent the table name and field name. We represent names that are more than one word, such as "COURSE_ID," as the mean of their word vectors. For out-of-vocabulary names, we use the vector for the "UNK" token. Whether a field is a primary key is represented with a single bit; the same is true for whether it is a foreign key. We represent its type using a one-hot vector. We concatenate all of these to form a single vector representing the field, as shown in Figure 6.2. We represent each table in the schema using a similar vector, except that the field name section is zeroed out. We then stack these vectors to generate a matrix representation of the entire schema.

A potential disadvantage of these schema embeddings is that they do not encode information about the relationships between fields in different tables. To remedy this, we also

[1] https://nlp.stanford.edu/projects/glove/

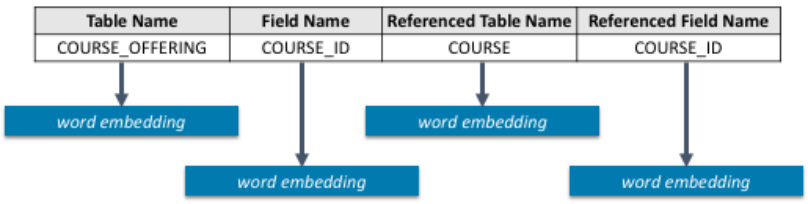


Figure 6.3: Schema map embedding for relationship of COURSE_OFFERING.COURSE_ID and COURSE.COURSE_ID.

create a schema map to represent foreign key constraints between tables. A foreign key constraint has four major components: the table name and field name of the foreign key, and the table name and field name of the field it references. For example, in the advising schema, COURSE_OFFERING.COURSE_ID refers to COURSE.COURSE_ID. This information might be useful as the model seeks to connect information in the COURSE_OFFERING and COURSE tables. Thus, for each foreign key constraint, we concatenate the word embeddings of these four components, as shown in Figure 6.3. We stack these vectors to form a matrix map of the schema.

6.3.2 Seq2Seq Model

A seq2seq model (Sutskever et al., 2014) is comprised of two recurrent neural networks (here, LSTMs), an encoder and a decoder. At each time step t, an LSTM cell processes the hidden vector from the previous step, $\mathbf{h}_{t-1}$, and an input vector from the current time step, $\mathbf{x}_t$. That is,

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{6.1}$$

where LSTM is the LSTM function of Hochreiter and Schmidhuber (1997). In the encoder, $\mathbf{x}_t$ is an embedding representing the t-th token of the input question; in the decoder, the embedding is the $t-1$-th output token for the query.[2]

We calculate attention to the input sequence (*i.e.*, the question) in a similar way to Bah-

[2]Multi-layer LSTMs function slightly differently, but we use only single-layer LSTMs for the present work.

danau et al. (2015). Let $|q|$ be the number of tokens in the input question. Then $\mathbf{h}_1, ...\mathbf{h}_{|q|}$ are the hidden vectors emitted by the encoder. Attention scores used by the decoder at timestep t are a vector $\mathbf{s}^t$. Each scalar s_k^t represents the normalized attention score for $\mathbf{h}_k$, the encoder's k-th hidden vector. We first calculate the non-normalized attention score

$$\alpha_k^t = \mathbf{v}_{att} \tanh(\mathbf{W}_0 \mathbf{h}_k + \mathbf{W}_1 \mathbf{h}_t) \tag{6.2}$$

where $\mathbf{W}_0$, $\mathbf{W}_1$, and $\mathbf{v}_{att}$ are learned parameters. We then normalize using softmax:

$$s_k^t = \frac{\exp\{\alpha_k^t\}}{\sum_{j=1}^{|q|} \exp\{\alpha_j^t\}} \tag{6.3}$$

Using the attention scores as weights for the hidden vectors, we calculate a context vector for timestep t:

$$\mathbf{c}^t = \sum_{k=1}^{|q|} s_k^t \mathbf{h}_k \tag{6.4}$$

In addition, our model incorporates an explicit representation of the database schema, as described in Section 6.3.1. Let M be the schema embedding matrix; it is comprised of rows $\mathbf{m}_0, \mathbf{m}_1, ...\mathbf{m}_r$, where r is the number of elements in the schema, and u represents the length of each vector. We calculate a scalar attention score a_n^t for the n-th field or table in the schema, similar to the attention score of Eqn. 6.2:

$$\beta_n^t = \mathbf{v}_{att}^{\beta} \tanh(\mathbf{W}_3 \mathbf{m}_n + \mathbf{W}_4 \mathbf{h}_t) \tag{6.5}$$

$$a_n^t = \frac{\exp\{\beta_n^t\}}{\sum_{l=1}^{r} \exp\{\beta_l^t\}} \tag{6.6}$$

From those scores, we generate the attention vector for the schema:

$$\mathbf{v}^t = \sum_{j=1}^{r} a_j^t \mathbf{m}_j \tag{6.7}$$

We then concatenate the current hidden vector, input context vector, and schema attention vector to get h_t^{att}. This vector is the input hidden vector for timestep $t+1$, and it is also used to predict the output at time t, such that

$$p(y_t|y_{<t}, q) = \text{softmax}(\tanh(\mathbf{W}_4(h_t^{att}))^\intercal \mathbf{e}(y_t) \tag{6.8}$$

Here, $\mathbf{e}(y_t)$ is simply a one-hot vector, enabling Eqn. 6.8 to specify the probability for y_t.

6.3.3 Attention-Based Copying

Attention-based copying from input sequences has been helpful in neural semantic parsing of English to logical forms (Jia and Liang, 2016). Here, we incorporate attention-based copying of both the input and the schema elements.

Specifically, our attention-based copying system has a vocabulary consisting of tokens that are part of the SQL language in general (*e.g.*, "SELECT", "FROM", "=", "LIKE"), as well as two special tokens, COPY_WORD and COPY_SCHEMA. When the LSTM cell output predicts a token from the SQL language, the model emits that token as usual. When the cell predicts COPY_WORD, the model emits the token from the input sequence with the highest attention score, as calculated by Eqn. 6.3.

When the cell predicts COPY_SCHEMA, the model determines the schema element to copy by considering the hidden state, the encoder context, the schema attention score, and weighted schema embeddings. To do this, we concatenate the hidden vector $\mathbf{h}_t$ and the context vector $\mathbf{c}^t$ from Eqn. 6.4. We apply a set of learned weights, $\mathbf{W}_5 \in \mathbb{R}^{u \times v+w}$, where u is the length of a schema embedding, v is the length of h, and w is the length of c, to the

schema embeddings. The schema attention copy score for the n-th schema element at time t is then

$$r_n^t = \beta_n^t * ([\mathbf{h}_t, \mathbf{c}_t] \cdot \mathbf{W}_5 \mathbf{m}_n) \tag{6.9}$$

Note that calculating the expression within parentheses does not depend on the number of elements in the schema.

If the decoder emits any non-COPY token at $t-1$, then the input to the decoder at timestep t, x_t, is the word embedding for that token, as it would be in a normal seq2seq model. If it emits COPY_WORD, then $x_t = \mathbf{h}_{\hat{k}}$, where

$$\hat{k} = \arg\max_k s_k^{t-1} \tag{6.10}$$

Similarly, if it emits COPY_SCHEMA, $x_t = \mathbf{W}_0 \mathbf{m}_{\hat{n}}$, where $\hat{n}$ maximizes a_n^{t-1}.

Loss is the sum of three components:

$$loss = l_{token} + l_w + l_s \tag{6.11}$$

l_{token} is the categorical cross entropy for the model's probability distribution over the output vocabulary tokens; in other words, the same loss as used in the model without attention-based copying.

l_w is the loss for word copying, and l_s is the loss for schema copying. Each is zero when the model should not copy from the specified source; otherwise, it is the categorical cross entropy of the distribution of scores over the tokens that might be copied. That is, if $H(p, q)$ is the categorical cross entropy of a distribution q and the true distribution p,

$$l_w = \begin{cases} H(\hat{s}^t, s^t) & \text{if } \hat{y}_t = COPY\_WORD \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

	Number of Questions	Number of Unique Queries
Advising	3898	208
ATIS	5280	947
GeoQuery	877	247
Scholar	817	193

Table 6.1: Text-to-SQL dataset sizes

where $\hat{y}_t$ is the correct token from output vocabulary at time t, $\hat{s}^t$ is a one-hot vector indicating the correct word from the input sequence to copy, and s^t is the vector of s_k^t values from Eqn. 6.3. Likewise,

$$l_s = \begin{cases} H(\hat{r}^t, r^t) & \text{if } \hat{y}_t = COPY\_SCHEMA \\ 0 & \text{otherwise} \end{cases} \tag{6.13}$$

where $\hat{r}^t$ is a one-hot vector indicating the correct token from the schema to copy, and r^t is actual vector of schema attention copy scores.

6.4 Datasets

As shown in Chapter 5, best practices for evaluating text-to-SQL systems call for evaluation on multiple datasets and multiple splits.

We evaluate using four datasets, each comprised of English questions and corresponding SQL queries. Advising (Chapter 5) is in the student-advising task domain. ATIS (Price, 1990; Dahl et al., 1994; Iyer et al., 2017) relates to a flight-booking task. GeoQuery (Zelle and Mooney, 1996; Popescu et al., 2003; Giordani and Moschitti, 2012; Iyer et al., 2017) contains questions and queries about United States geography. And Scholar (Iyer et al., 2017) is made of questions about academic papers, authors, and venues. All four datasets include at least some joins and nesting. Dataset sizes are described in Table 6.1. We use the standardized versions as described in Chapter 5.

Also in accordance with Chapter 5, we evaluate on both question-based splits, where no English question may appear in both train and test sets, and query-based splits, where no anonymized SQL query may appear in both train and test sets. As we have shown, query-based splits are far more difficult than traditional question-based splits.

We use the datasets in two different ways. In one, we train each model on the train set of one dataset and test it on the test set of the same dataset. This is the standard approach, and the only one reported on in all work we are aware of. In the other, we train one model on train sets from all four datasets and test on each dataset's test set. This give some indication of the ability of a model to generalize across domains. In one sense, it is a harder problem, since the model needs to learn not only how to generate SQL from text, but also how to classify which domain it is in. In another sense, however, it has the potential to make the problem easier, if the model can be taught to use what it learns about SQL in one domain to improve its performance in another. A model that can do this would be extraordinarily useful because it could be ported to domains that do not have much training data.

6.5 Experiments

We report on two sets of experiments.

For the first, we ask whether attention to schema embeddings and attention-based copying from schema can improve performance of a state-of-the-art system. We begin with a seq2seq model with attention-based copying from the input question, which was the strongest system in our experiments in Chapter 5. In the first condition, we add attention to schema. In the second, we add both attention to schema and attention-based copying from schema. We hypothesize that attention alone will have a small effect, and copying will have a larger effect. We expect the strongest improvement to be in the cross-domain model. There, schema attention should provide a strong signal of which domain a question comes from. Moreover, copying from schema should enable the model to generalize from

one schema to another.

For the second set of experiments, we investigate two representations of schemas. We report the results of a seq2seq model with attention to input and attention to schema embeddings. We compare this to the same model with attention to input, schema embeddings, and the schema map. We hypothesize that the additional information about relationships in the schema that the schema map provides should improve performance.

For both experiments, our metric is exact-match query accuracy. A query is correct if it is identical to the gold query; otherwise, it is counted as incorrect.

6.6 Results and Analysis

6.6.1 Schema Attention and Copying

	Advising		ATIS		GeoQuery		Scholar	
	Ques.	Query	Ques.	Query	Ques.	Query	Ques.	Query
Attn. Copying from Input	70	3	68	2	**76**	26	58	**11**
+ Trained on all	67	3	71	7	67	16	45	**11**
Attn to schema	67	2	**73**	2	67	**30**	**59**	7
+ Trained on all	63	3	68	**12**	71	14	50	7
Copying from both	**71**	3	62	2	63	20	3	3
+ Trained on all	63	3	64	0	47	13	3	0

Table 6.2: Accuracy of a seq2seq attention-to-input copying model with no schema input, with attention to schema, and with attention-based copying from schema on question and query splits of dev sets for the four datasets.

As can be seen in Table 6.2, experiments on whether attention to schema and copying from schema are helpful remain inconclusive. No model consistently outperformed other models. No model consistently outperformed other models on one split but not the other. In light of these results, we cannot reject the null hypothesis (i.e., that attention to schema and copying from schema do not affect model performance).

Copying from both on Scholar is an obvious outlier. Error analysis reveals that the

	Advising	ATIS	GeoQuery	Scholar
Schema attn.	73	90	69	85
Schema copying	81	79	50	43

Table 6.3: Percent overlap in incorrect questions between experimental systems and baseline. Wrong question overlap is the number of questions for which both the experimental and baseline system generated incorrect SQL divided by the total number of questions for which the experimental system generated incorrect SQL.

schema-copying model frequently copied the wrong table and field names. Omitting these errors, the model's performance would be comparable to the schema attention model. Interestingly, in Advising and ATIS, all table and field names are either a single word or multiple words separated by underscores (e.g., COURSE_ID). In GeoQuery, the same is true except for the HIGHLOW table. In Scholar, however, many table and field names are made of multiple words strung together with no delimiter between them, like PAPERID and PAPERKEYPHRASE. Such table and field names were not tokenized, meaning the word embedding portions of their schema embeddings were "UNK." The poor performance of the schema-copying model may thus be due to the number of UNK-based embeddings in the Scholar; the model has no way to tell the difference between the embedding for the PAPERDATASET table and that for the PAPERKEYPHRASE table.

Must we conclude that attention to schema and copying from schema make no difference, aside from the exceptional case of Scholar? By one measure, this seems to be true. The first two rows of Table 6.3 show the percentage of overlap between questions that the experimental models got wrong and questions that the baseline model got wrong. High numbers indicate that the experimental models primarily got the same questions wrong as the baseline model. Setting aside Scholar, we see that the same questions were generally difficult for all models.

Cross-domain training was neither consistently helpful nor consistently harmful. As expected, the baseline model does not seem to have the ability to generalize what it learns in one domain to another. Unfortunately, the experimental models also do no appear to generalize across domains. Models trained on all domains successfully classified unseen

	Advising		ATIS		GeoQuery		Scholar	
	Ques.	Query	Ques.	Query	Ques.	Query	Ques.	Query
Attn. to Input & Schema	**33**	**3**	**67**	2	**57**	**28**	**32**	3
+ Schema Map	29	2	65	2	53	23	28	**4**

Table 6.4: Accuracy of non-copying model with attention to input and schema embeddings, with and without attention to a schema map. Note: These results are currently trained on the train set and evaluated on the dev set.

questions into the correct domain; that is, we did not encounter errors where the model attempted to query the Advising database when the question came from Scholar. The models even appear able to distinguish the somewhat related ATIS and GeoQuery domains; even though both include references to cities and states, models trained on all datasets distinguish GeoQuery's STATE_ID field from ATIS's STATE_CODE field.

6.6.2 Schema Maps

Table 6.4 compares a model that pays attention to input and schema embeddings with one that pays attention to input, schema embeddings, and a schema map. The model without the schema map is more accurate. The differences are small, but the consistency of their direction indicates that use of a schema map hinders performance.

One would expect the schema map to have either a positive effect or none. After all, if it does not provide useful information, the model should learn to ignore its signal. The problem may be that the schema map makes the model larger, and the amount of training data we have cannot support a larger model.

6.7 Conclusions and Future Work

The proposed representations of schema and architectures that incorporate those representations have not proven helpful for the text-to-SQL task. However, the observation that SQL queries depend on the database schema remains true. Thus, additional work on incor-

porating schema information into neural networks for text-to-SQL could still be fruitful.

One option is to make changes to the architecture. Perhaps the attention-to-schema model could be improved with some form of gating. Or perhaps adjusting the structure to more closely resemble a pointer network would help.

The problem might lie with the schema representation. We use word embeddings to build these representations, but there is no reason to believe that these embeddings occupy the same space as the embeddings the model learns for the encoder or decoder vocabularies. We might address this by initializing all three with word embeddings from the same space and changing the schema embeddings from their current, static state to dynamically learned embeddings like those of the encoder.

None of these changes address the larger problem, however: text-to-SQL models are not learning to generalize. Within domains, models are failing to generalize to previously unseen queries. Across domains, models are not able to use information gleaned about SQL from one dataset to improve their performance on another dataset.

The most important future work will need to fix this. Encoding more knowledge of SQL into the network could help. This might be done through pre-training the parameters of the decoder as a SQL language model, perhaps on a dataset like that of Iyer et al. (2016). Such a model, it is to be hoped, would learn general rules of SQL, which could then be honed for particular domains. Alternatively (or additionally), we could explicitly provide the network with information about the structure of SQL. Zhong et al. (2017)'s seq2SQL architecture and Dong and Lapata (2016)'s seq2tree architecture are examples of this type of idea; however, to date no one has built an architecture that is both specific to SQL and general enough to cover a variety of queries. Other changes to both architecture and input data should focus on how to train the network to recognize and use compositionality. These are the areas most likely to generate true breakthroughs, rather than incremental improvements over the state of the art.

CHAPTER 7

Text-to-SQL in Dialog Context

7.1 Introduction

Text-to-SQL work to date has focused on transforming a single question—such as "Who teaches Discrete Mathematics?"—to a query (Popescu et al., 2003; Popescu et al., 2004; Giordani and Moschitti, 2012; Poon, 2013; Li and Jagadish, 2014; Saha et al., 2016; Zhong et al., 2017; Iyer et al., 2017; Cai et al., 2017). One potential use for parsing English to SQL is as a component of a dialog system. In a dialog system, an agent typically holds a multi-turn conversation with a user. Thus, the meaning of a given question may be affected by the conversational context.

An obvious example is coreference. Suppose a student and an advisor are having a conversation about what courses the student should take the following semester. Their conversation might include the following exchange:

ADVISOR: You should consider taking Discrete Mathematics.

STUDENT: Who teaches that class?

Notice that the student's question is semantically equivalent to "Who teaches Discrete Mathematics" when viewed in the context of the conversation; it can be answered using the same SQL query. However, the single utterance "Who teaches that class?" does not contain enough information to generate the SQL.

Conversational state may also be important. Suppose our student and advisor had the

following interaction:

STUDENT: I'm looking for an easy class.

ADVISOR: Have you considered EECS 484?

STUDENT: When does that meet?

ADVISORS: Monday mornings.

STUDENT: I can't do morning classes. Can you suggest something else?

"Can you suggest something else?" does not make its constraints explicit, but a human would know that the student wants easy classes that do not meet in the afternoon. A dialog system would need to maintain some representation of the conversation's state to remember that it needs to generate an easiness constraint.

In this chapter, we therefore take the first steps towards transforming the one-to-one text-to-SQL systems of Chapters 5 and 6 into text-to-SQL components for dialog systems. We describe the Flex-to-SQL dataset, version 0.1. We report the performance baseline systems on this dataset. And we describe future work that is promising for this task.

7.2 Dataset

Since no prior work has considered the dialog-snippet-to-SQL task, we describe a new dataset, Flex-to-SQL dataset, version 0.1, for the task. It combines data from the Flex Dialog dataset with the Advising SQL dataset.

The Flex Dialog dataset (Jiang et al., 2018 forthcoming) consists of dialogs between two university students role-playing an undergraduate student and academic advisor. The "student" received a made-up student profile, and the "advisor" received a set of courses recommended for the student, along with information about each course. The participants then interacted through a chat interface, holding a conversation with the goal of helping the "student" select courses for the following semester.

The Advising SQL dataset (Chapter 5) includes over 200 unique SQL queries corre-

sponding to questions an undergraduate computer science student might ask an advisor.

7.2.1 Preliminary Annotation

To develop the Flex-to-SQL dataset, researchers spent one day annotating Flex dialogs. After a training session on shared dialogs to ensure annotators understood the task, one annotator reviewed each dialog. Annotators highlighted each student utterance that represented a query that might be answered by a database. For each highlighted utterance, the annotator selected a label from a searchable list. Labels comprised either a question and the tag "CLOSEST" or "EXACTLY", or just the choice "OTHER."

Questions that could be used for the labels initially came from the Advising SQL dataset, with one representative question for each SQL query. Annotators also contributed to a shared list of questions that occurred often in the dialogs, and the searchable list was updated to include items on the list approximately once an hour.

Annotators labeled each highlighted utterance with a question from the dropdown that was as semantically similar to the utterance as possible. If the selected question was an exact paraphrase of the highlighted utterance (but for entity names), the "EXACTLY" tag was used; if it was not an exact match, but rather the closest match available in the list, the "CLOSEST" tag was used. If no question from the list was close in meaning to the highlighted utterance, annotators applied the "OTHER" label.

To keep the questions as broad as possible, named entities in the labels were replaced with variables. The label "Who teaches department0 number0 next semester?" thus applied equally to "Who teaches EECS 280 next semester?" and "Who teaches EECS 203 next semester?"

This annotation scheme had multiple goals. First, it enabled annotators who might not be familiar with SQL to quickly indicate whether an utterance was semantically equivalent to a SQL query already in the Advising SQL dataset. Second, for questions that did not already have SQL queries, it allowed rough clustering based on the nearest query.

Agree	The label is exactly right.
Closest	The label is close, but needs to be part of the "closest" review.
Unclear	The question is unclear and shouldn't generate SQL.
Data	The question asks for data that isn't in the database and shouldn't generate SQL.
Not a query	Annotator mistake; this was not a query at all.
Other	Explain in comments.

Table 7.1: Labels used in the second-level review of questions tagged "EXACTLY."

7.2.2 Subsequent Annotation of "Exact" Utterances

Two native English speakers familiar with the database schema performed a second level of review for every utterance tagged "EXACTLY." They were shown the dialog up to and including the tagged utterance (the "utterance in context"). They labeled each utterance with one of the options in Table 7.1. A comment field was available for all labels and required for any question labeled "Other."

After the second-level review, 408 utterances in context had been identified as semantically equivalent to a question in the Advising dataset. The remainder were either identified as exactly matching a question for which there was no SQL query yet, removed as either unclear or requesting data that was not available from the database, or transferred to the "CLOSEST" bucket for further review.

For the 408 utterances that exactly matched a question in the Advising dataset, we were able to create a dataset largely automatically. Each English question input was the entire utterance in context. That is, the "question" consists of the entire conversation up to and including the labeled utterance. Each query was the SQL query from Advising corresponding to the exact-match question. However, this still left the identification of variables in the utterance to be done manually.

As noted above, label questions included variables; for instance, "Who's the EECS 280 instructor next semester?" would be labeled "Who teaches department0 number0 next semester? EXACTLY" To run against the database and get correct results, we must replace "department0" with "EECS" and "number0" with "280" in the SQL. For these utterances,

we manually identified the variables.

The resulting dataset is Flex-to-SQL v.0.1. It contains 41 distinct SQL queries corresponding to the 408 utterances. We created a question-based split and a query-based split, though for the pilot work we report only experiments on the question-based split. Since the dataset is relatively small, we use cross validation. For the query-based split, we use leave-one-query-out cross-validation. For the question-based split, we randomly assigned each question to one of ten buckets.

7.3 Pilot Experiments

In light of Chapters 5 and 6, an obvious baseline for this task is the sequence-to-sequence (seq2seq) with attention-based copying model.

Our initial plan was to provide the utterance in context as the input to the encoder. However, due to the length of the context, this created an exploding or vanishing gradient problem.[1] For these pilot experiments, we therefore limited the length of context to at most 150 characters. We used the last 150 characters of each context, to ensure that the target utterance was included, and on the assumption that context closer to the target utterance would be more important than context earlier in the conversation.

In addition to training an input-copying seq2seq model on Flex-to-SQL itself, we report the performance of a model trained on the Advising dataset's question split, evaluated on all examples in Flex-to-SQL v.0.1. Since all of the queries in Flex-to-SQL v.0.1 appear in Advising, the model will not need to generalize to a new domain, nor even to new queries

[1] Although LSTMs are designed to reduce the exploding and vanishing gradient problem seen in RNNs, very long input sequences can still create this problem for LSTM encoders, as it did in this case. The loss quickly became too large or too small, generating NaN (not a number) errors and making any training impossible.

We first attempted to address the problem by adjusting hyperparameters. We decreased the learning rate from $1e-3$, which had shown the best performance across single-utterance text-to-SQL datasets. We gradually decreased it to $1e-12$, but this did not remedy the problem. In addition, we tried a smaller network (50 hidden units for encoder and decoder, compared to 200 typically used for other datasets). We also tried using a simple stochastic gradient descent (SGD) optmizer in place of the Adam optimizer that had performed well on other datasets. None of these adjustments solved the problem.

Model trained on	Accuracy
Advising	0
Flex-to-SQL	39

Table 7.2: Results of pilot experiments on a new dialog-to-SQL task

within the same domain, in order to be successful.

7.4 Results

A model that acheived over 70% accuracy on the Advising dataset was unable to generate a single correct output for dialogs in the same domain that use the same queries. This is strong evidence that text-to-SQL in dialog is a distinct problem that will not be solved even by perfecting performance on single-utterance text-to-SQL datasets.

Preliminary results training on dialogs show that the task is not entirely impossible, but will be difficult.

One factor that makes this task more difficult is that it requires the model to filter out extraneous information. For example, in the Advising dataset, any course number mentioned in the question is likely to appear in the query as well. In Flex-to-SQL, though, the context often includes many course numbers, and the model will need to learn which are irrelevant. Consider the following dialog:

STUDENT : are there any other courses i should take

ADVISOR : Taking EECS 370 with EECS 281 is a good choice .

STUDENT : what is EECS 370 ?

The gold query is

```
SELECT COURSEalias0.DEPARTMENT, COURSEalias0.NAME, COURSEalias0.NUMBER
FROM COURSE AS COURSEalias0
WHERE COURSEalias0.DEPARTMENT = "EECS" AND COURSEalias0.NUMBER = 370 ;
```

but the system outputs

```
SELECT COURSEalias0.DESCRIPTION, COURSEalias0.NAME
FROM COURSE AS COURSEalias0
WHERE COURSEalias0.DEPARTMENT = "EECS" AND COURSEalias0.NUMBER = 281 ;
```

7.5 Future Work

This is pilot work, and there is obviously room for a great deal of improvement. As the current errors show, exploring improved copying-from-input may be important.

Another problem to overcome is the length limitation. If the problem is in fact vanishing and not exploding gradients, a larger epsilon value might address it. Intelligent initialization of weights for the model might help with exploding gradients. If the weights are random, then costs at the beginning of training will be high. However, if the weights are initialized to something close to what we expect will work, the initial costs may be low enough to avoid this problem. One method of intelligent initialization to try would be training on only the shortest examples first before gradually permitting longer and longer examples in the training data.

Experimenting with other architectures would probably also be worthwhile. For example, instead of an LSTM encoder, what would happen if we used a convolutional neural network (CNN)? Or perhaps a combination would be helpful, with an LSTM over a small portion of the text and a CNN over a larger portion.

Alternatively, we can investigate other ways of providing information early in the conversation to the model without passing the entire context into the encoder. For instance, would a simple coreference resolver help to label entities in the target utterance, so that less context is needed?

On the data side, we intend to expand the available data for this task. First, we will annotate additional dialogs, generating new SQL queries for questions labeled “closest” in the first round of annotation. Second, many paraphrases of the dialogs already exist. We

can therefore compare a model trained on a dataset augmented with paraphrases without the need to annotate new dialogs.

7.6 Conclusion

In this chapter, we have taken the first steps on a new and important task, applying text-to-SQL methods to dialogs. We have shown that generalizing from single-utterances to dialogs is non-trivial. We have described the first version of a new dataset for this task. We have established baseline results and proposed a number of methods of improving performance.

CHAPTER 8

Conclusion

This work has studied computational meaning representations (MRs) for NLP from two perspectives: What MRs are appropriate for what tasks, and how should we generate our selected MR from text?

An enormous number of options for MRs exist, ranging from simple word counts to lambda calculus to SQL to neural networks. We can even choose to use natural language as its own representation. The lesson of Part I is about the tradeoffs between different MRs for different tasks.

Natural language as its own MR has a number of advantages: It's interpretable even by lay-people. It is guaranteed to be expressive enough to cover anything that was expressed in natural language. And it saves us the trouble of generating an MR. This is not just convenient; it removes a step in the pipeline for any NLP task where things can go wrong.

Extractive summarization is one of the most marked success stories for natural language as its own MR. While it uses other MRs (such as tf*idf) to determine which sentences to extract, it retains the sentences from the original document as their own MR when generating the output summary. However, when we change the task a little, we see a major downside of natural language as its own MR; namely, it is very difficult to manipulate. We used two state-of-the-art methods as well as our own set of hand-engineered rules to try to shorten long sentences, and we found that this introduced so many grammatical errors as to make the text unreadable. In essence, the effort that is saved by not converting text

to a different MR is made up for by the effort required to manipulate the text directly; the possible errors that we seek to avoid by skipping the MR-generation step are not avoided, but shifted to a different step in the pipeline.

Neural networks have shown incredible promise for NLP tasks. Looking at recent NLP conference proceedings, it is difficult to avoid concluding that LSTMs are a magical MR that will solve all of the open problems in the field; one need only tweak the architecture just so for the task at hand. In Chapter 3, however, we saw that neural networks are not always the answer. We tried three LSTM network structures, and none of them performed as well as a state-of-the-art system that uses an SVM. Neural networks may be an effective hammer for NLP, but not every application is a nail.

One lesson to take from these two chapters is that choice of MR is important, and deciding what MR to use for a particular purpose is a worthwhile subject of research in itself. Chapter 4 reflects this lesson: we set out to compare two types of representations for short text clustering. Our hypothesis that distributional-semantics-based MRs would have an advantage on texts that express creativity through a wider vocabulary acknowledges that MRs are not one-size-fits-all. And, indeed, this work makes that point twice: once by showing which MRs worked best for creative and not-so-creative datasets, and a second time by showing the unexpected interaction between the choice of MR and the choice of clustering algorithm.

What, then, can we conclude about selecting MRs for short texts? Simply that it is a complex issue. Properties of the dataset, the task, and other stages of the pipeline it will be used in all play a role in choosing the right tool for the job.

In Part II, we assumed that, after taking all of this into account, SQL was the right MR for the job. We then asked questions about generating SQL from text. While we approached this problem from three different angles—methodological, systems, and applications—a single recurring theme emerged from all three: generalization.

When we looked at the text-to-SQL task from a methodological point of view, we asked

whether there was room for improvement in evaluation methods. We found two such areas. The first is in our choice of datasets: if we want to determine whether text-to-SQL systems perform well on human-generated questions over real relational databases, then we need to evaluate on datasets that include joins and nesting.

The second area where text-to-SQL evaluation methods can be improved is in how datasets are split. The traditional split, based on English questions, allows a classifier with a slot-filler to achieve performance on par with state-of-the-art systems. The fact that such a system, with no ability to generalize to new queries, can succeed on the task suggests that our measure of success does not include a measure of generalizability. Splitting the data based on SQL queries instead makes the problem much more difficult, but also enables us to study systems' ability to generalize to previously unseen queries.

In Chapter 6, we set out to improve a seq2seq model for text-to-SQL. We sought to provide explicit information about the schema of the database being queried, to allow our system to learn a general model of text-to-SQL and consult the schema for details of a specific database. Our hope was that this would not only improve performance on a single dataset, but also improve the ability to generalize across domains. Unfortunately, our model did not benefit from attention to the schema representation we used for these experiments.

More work will be needed to determine how schema representations can be used for this task. There are, of course, plenty of areas for further experimentation. We can change the architecture of the network, including incorporation of gates or a more pointer-network-like structure. We can tweak our schema representation and our embedding space. However, rather than random experimentation, we should make sure we focus on what will enable us to learn a general model of SQL and use the schema embeddings to apply that model to specific domains. This could mean structuring the network in ways that force the model to output valid SQL, or pretraining a decoder on a large corpus of SQL, or any number of other techniques, provided they are guided by our goal of generalizability.

Finally, we discovered that text-to-SQL techniques for single-utterance questions do

not generalize well to dialog. All previous text-to-SQL work has focused on improving performance on single-sentence questions. When we take a system trained on such data, however, and apply it to snippets of dialog in the same domain, it is incapable of generating correct output. Moreover, when a state-of-the-art text-to-SQL system is trained instead on a dialog-to-SQL dataset, its accuracy is cut in half. Dialog-to-SQL is thus a new task. Future work will need to acknowledge this, either by using different methods than single-utterance text-to-SQL or by improving the generalizability of single-utterance systems.

Part II thus reveals a difficult but important goal for text-to-SQL research: building systems that generalize. Whatever angle we approach the problem from, this stands out as an area ripe for improvement.

In summary, the contributions of this thesis are

- Exposure of the difficulty even state-of-the-art sentence-shortening methods when applied in a summarization pipeline for sophisticated documents;
- Experimental results for three LSTM architectures applied to paraphrase detection;
- Guidelines for choosing MRs and clustering algorithms for short text clustering;
- Best practices for evaluating text-to-SQL systems, as well as identification of properties future datasets for this task should include;
- Experimental results for seq2seq models with attention to two different representations of schemas on the text-to-SQL task; and
- Discovery of the distinction between single-utterance text-to-SQL and text-to-SQL in a dialog context.

And the most relevant future directions identified in this work are

- For sentence-shortening in sophisticated document summaries, a seq2seq model trained on Wikipedia and Simple English Wikipedia is an obvious next step. Before using it

for summarization, pilot work should check for grammatical output that still captures the original meaning of the sentence.

- For paraphrase detection using tree-LSTMs, incorporating dependency labels might improve performance.
- Schema-attention models for text-to-SQL might benefit from
  - Modifications to network architecture, such as use of additional gating, trees in place of sequences for encoder input, or decoder networks built around the grammar of SQL.
  - Improvements in schema representations, such as dynamically updating schema embeddings, or encoder embeddings in the same vector space as the schema embeddings.
  - Incorporating greater knowledge of SQL grammar into the decoder, either expressly or by pretraining a language model on a large SQL dataset.
- Dialog-to-SQL is a new problem that will benefit from expansion of available resources.
- Future work on dialog-to-SQL should also compare an end-to-end system against systems that attempt to summarize relevant points of the dialog into a single utterance, as well as measure how much context before the question ought to be included for end-to-end models.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic Parsing as Machine Translation. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52.

I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. 1995. Natural Language Interfaces to Databases - An Introduction. *Natural Language Engineering*, 1(709):29–81.

Gabor Angeli, Neha Nayak, and Christopher D Manning. 2016. Combining Natural Logic and Shallow Reasoning for Question Answering. In *ACL*, pages 442–452.

Philip Arthur, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Semantic Parsing of Ambiguous Input through Paraphrasing and Verification. In *Transactions of the Association for Computational Linguistics*, volume 3, pages 571–584.

Yoav Artzi and Luke S Zettlemoyer. 2011. Bootstrapping Semantic Parsers from Conversations. *Computational Linguistics*, pages 421–432.

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

Yoav Artzi, Nicholas FitzGerald, and Luke Zettlemoyer. 2013. Semantic Parsing with Combinatory Categorial Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Tutorials)*, page 2.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710.

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II*.

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. 2014. Exploiting discourse information to identify paraphrases. *Expert Systems with Applications*, 41(6):2832–2841.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the ICLR*, pages 1–15, San Diego, California.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. 2007. Clustering short texts using wikipedia. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 787–788. ACM.

Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, volume 17. Association for Computational Linguistics.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Abdullah Bawakid and Mourad Oussalah. 2008. A semantic summarization system: University of Birmingham at TAC 2008. *Tac*.

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Semantic parsing using distributional semantics and probabilistic logic. *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 7–11.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1415–1425.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. *Proceedings of EMNLP*, pages 1533–1544.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

Pavel Berkhin. 2006. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-Coverage Semantic Representations from a CCG Parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, pages 1240–1246.

Denny Britz, Anna Goldie, Minh-thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *ArXiv e-prints*.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Elia Bruni, Giang Binh Tran, and Marco Baroni. 2011. Distributional semantics from text and images. *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 22–32.

Fan Bu, Hang Li, and Xiaoyan Zhu. 2012. String re-writing kernel. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 449–458. Association for Computational Linguistics.

Qingqing Cai and Alexander Yates. 2013. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433.

Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, and Zijian Li. 2017. An encoder-decoder framework translating natural language to database queries. *CoRR*, abs/1711.0.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of English newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10.

Yllias Chali and Sadid A Hasan. 2012. On the effectiveness of using sentence compression models for query-focused multi-document summarization. *Proceedings of COLING 2012*, pages 457–474.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3):183–190.

Mao Chen, Klaus Dorer, Ehsan Foroughi, Fredrik Heintz, Zhanxiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Jan Murray, Itsuki Noda, Oliver Obst, Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. 2003. RoboCup Soccer Server 7.07, Manual. Technical report, The {R}obo{C}up Federation.

Bo Chen, Le Sun, Xianpei Han, and Bo An. 2016. Sentence Rewriting for Semantic Parsing. *Acl*, pages 766–777.

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1531–1542, Lisbon, Portugal, September. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *EMNLP-CoNLL*, pages 1–11.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *J. Artif. Intell. Res.(JAIR)*, 31:399–429.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

James Clarke, Dan Goldwasser, Ming-wei Chang, and Dan Roth. 2010. Driving Semantic Parsing from the World's Response. *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriber. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. *Proceedings of the workshop on Human Language Technology*, pages 43–48.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An Incremental Parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536—-546. Association for Computational Linguistics.

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. NA Smith. 2010. Probabilistic frame-semantic parsing. *HLT '10 Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 3(June):948–956.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-Semantic Parsing. *Computational Linguistics*, 40(1):9–56.

Hal Daumé, III and Daniel Marcu. 2005. Bayesian summarization at duc and a suggestion for extrinsic evaluation. In *Proceedings of the Document Understanding Conference, DUC-2005, Vancouver, USA*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

William Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*. International Conference on Computational Linguistics, August.

William E. Donath and Alan J. Hoffman. 1973. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1:33–43.

Güneş Erkan and Dragomir R. Radev. 2004a. LexRank : Graph-based Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Günes Erkan and Dragomir R. Radev. 2004b. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December.

Marcelo Luis Errecalde, Diego Alejandro Ingaramo, and Paolo Rosso. 2010. A new anttree-based algorithm for clustering short-text corpora. *Journal of Computer Science & Technology*, 10.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Yansong Feng and Mirella Lapata. 2010. Visual Information in Semantic Representation. *In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, Los Angeles, California, June 2010*, 9(June):91–99.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52. Citeseer.

Charles J. Fillmore, Christopher R. Johnson, and Miriam R L Petruck. 2003. Background to framenet. *International Journal of Lexicography*, 16(3):235–250.

Andrew Finch, Young-Sook Hwang, and Eiichiro Sumita. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, pages 17–24.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using Tree Transducers. *Proceedings of NAACL*, pages 731–739.

Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science*, 315(5814):972–976.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.

Ruifang Ge and Raymond J Mooney. 2009. Learning a Compositional Semantic Parser using an Existing Syntactic Parser. In *Proceedings of the ACL-IJCNLP*, pages 611—-619.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

Daniel Gildea. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to SQL queries with generative parsers discriminatively reranked. In *COLING 2012*, pages 401–410.

Dan Goldwasser and Roi Reichart. 2011. Confidence driven unsupervised semantic parsing. *HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 1:1486–1495.

David Graff and Christopher Cieri. 2003. English Gigaword LDC2003T05.

Edward Grefenstette, Phil Blunsom, Nando De Freitas, and Karl Moritz Hermann. 2014. A Deep Architecture for Semantic Parsing. *ACL Workshop on Semantic Parsing*, pages 22–27.

Gregory Grefenstette. 1998. Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind. In *Working notes of the AAAI Spring Symposium on Intelligent Text summarization*, pages 111–118.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 864–872. Association for Computational Linguistics.

Ben Hachey and Claire Grover. 2005. Automatic Legal Text Summarisation: Experiments with Summary Structuring. *Proceedings of the 10th international conference on Artificial intelligence and Law*, pages 75–84.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370. Association for Computational Linguistics.

Thrse Firmin Hand. 1997. A proposal for task-based evaluation of text summarization system. In *Proceedings of the Association for Computational Linguistics / European Association for Computational Linguistics Summarization Workshop*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, Lisbon, Portugal. Association for Computational Linguistics.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1011–1019, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Eduard Hovy and Chin-Yew Lin. 1998. Automated text summarization and the summarist system. In *Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998*, pages 197–214. Association for Computational Linguistics.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification*, 2(1):193–218.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Aminul Islam and Diana Inkpen. 2009. Semantic similarity of short texts. *Recent Advances in Natural Language Processing V*, 309:227–236.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, Berlin, Germany, August. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963—-973, Vancouver, Canada.

Rahul Jha, Catherine Finegan-Dollak, Reed Coke, Ben King, and Dragomir Radev. 2015. Content models for survey generation: A factoid-based evaluation. *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, 1:441–450.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *EMNLP*, pages 891–896.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Youxuan Jiang, Jonathan K. Kummerfeld, and Walter S. Lasecki. 2017. Understanding Task Design Trade-offs in Crowdsourced Paraphrase Collection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 103–109, Vancouver, Canada.

Youxuan Jiang, Jonathan K. Kummerfeld, Catherine Finegan-Dollak, and Walter S. Lasecki. 2018 (forthcoming). Effective crowdsourcing for a new type of summarization task. In *Proceedings of NAACL*, New Orleans, Louisiana.

Anders Johannsen, Héctor Martínez Alonso, and Anders Søgaard. 2015. Any-language frame-semantic parsing. In *Proceedings of EMNLP*, pages 2062–2066.

Siddhartha Jonnalagadda and Graciela Gonzalez. 2010. Sentence simplification aids protein-protein interaction extraction. *arXiv preprint arXiv:1001.4273*.

Rohit J. Kate, Yuk Wah Wong, and R.J. Raymond J. Mooney. 2005. Learning to transform natural to formal languages. *Proceedings Of The National Conference On Artificial Intelligence*, 20(3):1062–1068.

Tom Kenter and Maarten de Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM.

Tom Kenter, Alexey Borisov, and Maarten De Rijke. 2016. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. *Acl*, pages 941–951.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Applied Soft Computing Journal*, 30:737–747.

Ben King, Rahul Jha, Dragomir R Radev, and Robert Mankoff. 2013. Random walk factoid annotation for collective discourse. In *Proceedings of the 51st annual meeting on Association for Computational Linguistics*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284.

Beata Beigman Klebanov, Kevin Knight, and Daniel Marcu. 2004. Text simplification for information-seeking applications. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 735–747. Springer.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic Parsing with Semi-Supervised Sequential Autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087.

Zornitsa Kozareva and Andrés Montoyo. 2006. Paraphrase identification on the basis of supervised machine learning techniques. In *Advances in natural language processing*, pages 524–533. Springer.

Jayant Krishnamurthy and Tom M Mitchell. 2014. Joint Syntactic and Semantic Parsing with Combinatory Categorial Grammar. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1188–1198.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Q Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 957–966.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing Probabilistic CCG Grammars from Logical Form with Higher-Order Unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233.

Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical Generalization in CCG Grammar Induction for Semantic Parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*, pages 1512–1523.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-fly Ontology Matching. In *Emnlp*, pages 1545–1556.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014*, 32:1188–1196.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. In *Proceedings of the VLDB Endowment*, pages 73–84.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *EMNLP*, pages 490–500.

Junhui Li, Muhua Zhu, Wei Lu, and Guodong Zhou. 2015. Improving Semantic Parsing with Enriched Synchronous Context-Free Grammar. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, volume 1, pages 1455–1465.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 590–599, Portland, Oregon.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *ArXiv e-prints*, pages 1–7.

Chin-Yew Lin. 2003. Improving summarization performance by sentence compression: a pilot study. In *Proceedings of the sixth international workshop on Information retrieval with Asian languages-Volume 11*, pages 1–8. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *Naacl-2015*, pages 1076–1085.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Chenglong Ma, Weiqun Xu, Peijia Li, and Yonghong Yan. 2015. Distributional representations of words for short text classification. In *Proceedings of NAACL-HLT*, pages 33–38.

Shuming Ma, Xu Sun, Jingjing Xu, Houfeng Wang, Wenjie Li, and Qi Su. 2017. Improving Semantic Relevance for Sequence-to-Sequence Learning of Chinese Social Media Text Summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 635–640.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing - RTE '07*, page 193.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Inderjeet Mani and Mark T. Maybury, editors. 1999. *Advances in Automatic Text Summarization*. MIT Press, Cambridge, Massachusetts, USA.

Marco Marelli, Luisa Bentivogli, and Marco Baroni. 2014a. Task Description: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment - SemEval-2014 Task 1. In *SemEval 2014*, pages 1–8.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, pages 216–223. Citeseer.

André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 1–9. Association for Computational Linguistics.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2013. Learning to Parse Natural Language Commands to a Robot Control System. In *Experimental Robotics*, pages 403–415. Springer.

Kathleen McKeown, Rebecca J. Passonneau, David K. Elson, Ani Nenkova, and Julia Hirschberg. 2005. Do summaries help? In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217. ACM.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4. Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Advances In Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719, Doha, Qatar, October. Association for Computational Linguistics.

Dipendra K. Misra and Yoav Artzi. 2016. Neural Shift-Reduce CCG Semantic Parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1775—-1786.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Tom M. Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapa Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-Ending Learning. *AAAI Conference on Artificial Intelligence*, pages 2302–2310.

Arindam Mitra and Chitta Baral. 2016. Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning. *Proceedings of the 30th Conference on Artificial Intelligence (AAAI 2016)*, pages 2779–2785.

Andrew H Morris, George M Kasper, and Dennis A Adams. 1992. The effects and limitations of automated text condensing on reading comprehension performance. *Information Systems Research*, 3(1):17–35.

Gabriel Murray, Thomas Kleinbauer, Peter Poller, Tilman Becker, Steve Renals, and Jonathan Kilgour. 2009. Extrinsic summarization evaluation: A decision audit task. *ACM Transactions on Speech and Language Processing (TSLP)*, 6(2):2.

Preslav Nakov, Doris Hoogeveen, Lluís Arquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 27–48.

Arvind Neelakantan, Quoc V Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *Proceedings of the ICLR*, pages 1–10.

Preksha Nema, Mitesh Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven Attention Model for Query-based Abstractive Summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1063–1072.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4(2), May.

Mark E. J. Newman. 2004. Analysis of weighted networks. *Physical Review E*, 70(5):056131.

Mark E. J. Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582.

Mark Newman. 2010. *Networks: an introduction*. Oxford University Press.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press.

Rik Van Noord and Johan Bos. 2017. Dealing with Co-reference in Neural Semantic Parsing. In *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*.

Jahna Otterbacher, Dragomir Radev, and Omer Kareem. 2008. Hierarchical summarization for delivering information to mobile devices. *Information Processing & Management*, 44(2):931–947.

Shiyan Ou, Christopher S.G. Khoo, and Dion H. Goh. 2007. Automatic multidocument summarization of research abstracts: Design and user evaluation. *Journal of the American Society for Information Science and Technology*, 58(10):1419–1435.

Ankur P. Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded Semantic Parsing for Complex Knowledge Extraction. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 756–766.

Rodolfo A. Pazos Rangel, Juan Javier González Barbosa, Marco Antonio Aguirre Lam, José Antonio Martínez Flores, and Héctor J Fraire Huacuja. 2013. Natural language interfaces to databases: An analysis of the state of the art. In Oscar Castillo, Patricia Melin, and Janusz Kacprzyk, editors, *Recent Advances on Hybrid Intelligent Systems*, pages 463–480. Springer Berlin Heidelberg, Berlin, Heidelberg.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Henry Petersen and Josiah Poon. 2011. Enhancing short text clustering with small external repositories. In *Proceedings of the Ninth Australasian Data Mining Conference-Volume 121*, pages 79–90. Australian Computer Society, Inc.

B. Piwowarski, M.R. Amini, and M. Lalmas. 2012. On using a quantum physics formalism for multidocument summarization. *Journal of the American Society for Information Science and Technology*, 63(5):865–888.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 933–943.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces IUI 03*, pages 149–157.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 141–147.

Patti J. Price. 1990. Evaluation of spoken language systems: The ATIS domain. *Proc. of the Speech and Natural Language Workshop*, pages 91–95.

Vahed Qazvinian and Dragomir R. Radev. 2011. Learning from collective human behavior to introduce diversity in lexical choice. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1098–1108, Stroudsburg, PA, USA. Association for Computational Linguistics.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26. Association for Computational Linguistics.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 878–888.

Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the Special Issue on Summarization. *Computational Linguistics*, 28(4):399–408.

Dragomir R. Radev, Amanda Stent, Joel R. Tetreault, Aasish Pappu, Aikaterini Iliakopoulou, Agustin Chanfreau, Paloma de Juan, Jordi Vallmitjana, Alejandro Jaimes, Rahul Jha, and Robert Mankoff. 2015. Humor in collective discourse: Unsupervised funniness detection in the new yorker cartoon caption contest. *CoRR*, abs/1506.08126.

Daniel Ramage, Anna N. Rafferty, and Christopher D. Manning. 2009. Random walks for text semantic similarity. In *Proceedings of the 2009 workshop on graph-based methods for natural language processing*, pages 23–31. Association for Computational Linguistics.

Aniket Rangrej, Sayali Kulkarni, and Ashish V. Tendulkar. 2011. Comparative study of clustering techniques for short text documents. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 111–112, New York, NY, USA. ACM.

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé Iii. 2017. Biomedical event extraction using Abstract Meaning Representation. *Proceedings of the BioNLP 2017 workshop*, pages 126–135.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147. Association for Computational Linguistics.

Sarah Rastkar, G Murphy, and Gabriel Murray. 2014. Automatic summarization of bug reports. *IEEE Transactions on Software Engineering*.

G. J. Rath, A Resnick, and T. R. Savage. 1961. The formation of abstracts by the selection of sentences. part i. sentence selection by men and machines. *American Documentation*, 12(2):139–141.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the ACL*, 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the ACL*, 4:127–140.

Lawrence H. Reeve, Hyoil Han, and Ari D. Brooks. 2007. The use of domain-specific concepts in biomedical text summarization. *Information Processing & Management*, 43(6):1765–1776.

Samuel Ritter, Marco Baroni, Matthew Botvinick, Denis Paperno, and Adele Goldberg. 2015. Leveraging Preposition Ambiguity to Assess Compositional Distributional Models of Semantics. In *SemEval2015*, pages 199–204.

Stephen Roller and Sabine Schulte im Walde. 2013. A Multimodal LDA Model Integrating Textual, Cognitive and Visual Modalities. In *Emnlp*, pages 1146–1157.

Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. 2011. Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM*.

Vasile Rus, Philip M McCarthy, Mihai C Lintean, Danielle S McNamara, and Arthur C Graesser. 2008. Paraphrase identification with lexico-syntactic graph subsumption. In *FLAIRS conference*, pages 201–206.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Mrinmaya Sachan and Eric P Xing. 2016. Machine Comprehension using Rich Semantic Representations. *Acl 2016*, pages 486–492.

Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Ozcan. 2016. ATHENA : An Ontology-Driven System for Natural Language Querying over Relational Data Stores. In *Proceedings of the VLDB Endowment*, volume 9, pages 1209–1220.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 441–448. Association for Computational Linguistics.

Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. Technical report.

Prajol Shrestha, Christine Jacquin, and Béatrice Daille. 2012. Clustering short text and its evaluation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7182 LNCS(PART 2):169–180.

Advaith Siddharthan and Angrosh Mandya. 2014. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 722–731.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*, page 896.

Advaith Siddharthan. 2002. An architecture for a text simplification system. In *Language Engineering Conference, 2002. Proceedings*, pages 64–71. IEEE.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Advaith Siddharthan. 2010. Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 125–133. Association for Computational Linguistics.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D. Manning, and Andrew Y. Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Cliff C. Lin, Chris Manning, and Andrew Y. Ng. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural Headline Generation on Abstract Meaning Representation. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated Construction of Database Interfaces: Integrating Statistical and Relational Learning for Semantic Parsing. *Proceedings of the Joint SIGDAT Conference on Emprical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Machine Learning: ECML 2001*, pages 466–477.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*.

Kapil Thadani. 2014. Approximation strategies for multi-structure sentence compression. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1241–1251, Baltimore, Maryland, June. Association for Computational Linguistics.

Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond J Mooney. 2016. Learning Multi-Modal Grounded Linguistic Semantics by Playing " I Spy ". In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 3477—-3483.

Jesse Thomason, Jivko Sinapov, and Raymond J Mooney. 2017. Guiding Interaction Behaviors for Multi-modal Grounded Language Learning. In *Proceedings of the First Workshop on Language Grounding for Robotics*, pages 20–24.

Yun Tian, Haisheng Li, Qiang Cai, and Shouxiang Zhao. 2010. Measuring the similarity of short texts by word similarity and tree kernels. In *Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on*, pages 363–366. IEEE.

P.M. Tiersma. 1999. *Legal Language*. University of Chicago Press.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 290–297. Association for Computational Linguistics.

Z. Ul-Qayyum and W. Altaf. 2012. Paraphrase identification using semantic heuristic features. *Research Journal of Applied Sciences, Engineering and Technology*, pages 4894–4904.

Stijn Van Dongen. 2000. *Graph Clustering by Flow Simulation*. Ph.D. thesis, University of Utrecht.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606–1618.

Subhashini Venugopalan, Marcus Rohrbach, Trevor Darrell, Jeff Donahue, Kate Saenko, and Raymond Mooney. 2015. Sequence to Sequence Video to Text. In *International Conference on Computer Vision*, pages 4534–4542.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:3156–3164.

Andreas Vlachos and Stephen Clark. 2014. A new corpus for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–559.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the para-farce out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.

Jun Wang, Yiming Zhou, Lin Li, Biyun Hu, and Xia Hu. 2009a. Improving short text clustering performance with keyword expansion. In *The Sixth International Symposium on Neural Networks (ISNN 2009)*, pages 291–298. Springer.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009b. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 187–194, New York, NY, USA. ACM.

Adrienne Wang, Tom Kwiatkowski, and Luke S. Zettlemoyer. 2014a. Morpho-syntactic Lexical Generalization for CCG Semantic Parsing. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1284–1295.

Yu Wang, Lihui Wu, and Hongyu Shao. 2014b. Clusters merging method for short texts clustering. *Open Journal of Social Sciences*, 2(09):186.

Peng Wang, Jiaming Xu, Bo Xu, Cheng-Lin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. 2015. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 352–357.

Matthijs J. Warrens. 2008. On the equivalence of cohen's kappa and the hubert-arabie adjusted rand index. *Journal of Classification*, 25(2):177–183.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for Semantic Parsing with Statistical Machine Translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446.

Yuk Wah Wong and Raymond J Mooney. 2007. Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 960–967.

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 25–30. Association for Computational Linguistics.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 62–69.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Type- and content-driven synthesis of SQL queries from natural language. *ArXiv e-prints*.

Xiaohui Yan, Jiafeng Guo, Shenghua Liu, Xue-qi Cheng, and Yanfeng Wang. 2012. Clustering short text using ncut-weighted non-negative matrix factorization. In *Proceedings*

of the 21st ACM international conference on Information and knowledge management, pages 2259–2262. ACM.

Christopher C. Yang and Fu Lee Wang. 2008. Hierarchical summarization of large documents. *Journal of the American Society for Information Science and Technology*, 59(6):887–902.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China.

Wenpeng Yin and Hinrich Schütze. 2015a. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, Denver, Colorado, May–June. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2015b. Discriminative phrase embedding for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1368–1373, Denver, Colorado, May–June. Association for Computational Linguistics.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural Enquirer: Learning to Query Tables. *arXiv*, pages 1–17.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural Enquirer: Learning to query tables in natural language. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 2308–2314.

Jie Yin. 2013. Clustering microtext streams for event identification. In *IJCNLP*, pages 719–725.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*, 43(6):1549–1570.

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database queries using inductive logic proramming. In *Learning*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form : Structured Classification with Probabilistic Categorial Grammars. In *21st Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Computational Linguistics*, pages 678–687.

Yitao Zhang and Jon Patrick. 2005. Paraphrase identification by text canonicalization. In *Proceedings of the Australasian language technology workshop*, volume 2005, pages 160–166.

Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering relations between named entities from a large raw corpus using tree similarity-based clustering. In *Natural Language Processing–IJCNLP 2005*, pages 378–389. Springer.

Kai Zhao and Liang Huang. 2015. Type-Driven Incremental Semantic Parsing with Polymorphism. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 0041(2):1416–1421.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *ArXiv e-prints*, pages 1–12.